Social Signal Interpretation

Building Online Systems for Multimodal Behaviour Analysis

DISSERTATION

zur Erlangung des akademischen Grades
Doktor der Informatik

Lehrstuhl für Multimodale Mensch-Technik Interaktion
Universität Augsburg

M.Sc. Johannes Wagner

2015

Datum der Disputation	Augsburg, 21.12.2015
Erstgutachter	Prof. Dr. Elisabeth André
Zweitgutachter	Prof. Dr. Björn Schuller
Drittgutachter	Prof. Dr. Bernhard Möller

Zusammenfassung

In den vergangen Jahren hat sich *Social Signal Processing* (SSP) zunehmend als Forschungsgebiet im Bereich der Mensch-Maschine Interaktion etabliert. Ziel ist der Entwurf einer neuen Generation „sozialer Computer“, die im Umgang als natürlicher, effektiver und vertrauensvoller wahrgenommen werden. Bei einem Großteil der Arbeiten handelt es sich bisher allerdings um offline Studien, die soziale Interaktion unter Laborbedingungen untersuchen und damit oft ein übertrieben optimistisches Bild über die tatsächliche Anwendbarkeit vermitteln, da sie Probleme ausblenden die erst unter realen Bedingungen auftreten. Dies hat zur Folge, dass es trotz umfangreicher Anstrengungen bis heute nur wenige wirkliche Anwendungen gibt.

Die vorliegende Arbeit soll Entwickler motivieren mehr Anstrengungen in onlinefähige Systeme zu investieren. Zu diesem Zweck werden Defizite bisheriger Arbeiten analysiert und methodische und technische Lösungen präsentiert, die den Weg hin zu einer stärker anwendungsorientierten Entwicklung ermöglichen. So sind die meisten heute verfügbaren SSP Korpora noch immer audiovisuell und setzen sich aus isolierten und überzogen exemplarischen Beispielen zusammen. Um die Aufzeichnung neuer Datensätze zu erleichtern und mit zusätzlichen Messungen anzureichern, wird ein Mechanismus vorgeschlagen, der es erlaubt audiovisuelle Daten mit weiteren Eingabesignalen wie Körperbewegung, Blickverhalten, und physiologischen Reaktionen zu synchronisieren. Um die Information aus derart vielfältigen Quellen zu kombinieren, braucht es aber auch intelligente Fusionsstrategien. Leider zeigt sich, dass herkömmliche Ansätze in realistischen Szenarien keine zufriedenstellende Performanz liefern. Es wird deshalb ein neuer Fusionsansatz eingeführt, der es erlaubt die zeitlichen Zusammenhänge zwischen Modalitäten besser zu berücksichtigen. Auch ist es gängige Praxis die Datensätze in offline Studien für den Klassifikationsprozess zu optimieren, z. B. Abschnitte mit wenig Interaktion oder nicht-prototypischem Verhalten zu entfernen. Falls die Konditionen während der Lernprozesse jedoch zu stark von den späteren Anforderungen abweichen, kann dies zu suboptimalem Verhalten in der Anwendung führen. Es wird deshalb ein stärker anwendungsorientierter Ansatz vorgeschlagen.

Um den Mehraufwand bei der Entwicklung von echtzeitfähigen Systemen zu minimieren, braucht es Werkzeuge, die dem Entwickler möglichst viel Arbeit abnehmen. Aus diesem Grund wird ein frei verfügbares Framework namens SOCIAL SIGNAL INTERPRETATION (SSI) vorgestellt, das alle Schritte des maschinellen Lernens unterstützt und die Erstellung komplexer Vearbeitungspipelines aus einzelnen, wiederverwendbaren Komponenten ermöglicht. Die genannten Synchronisierungs- und Fusionsverfahren wurden mit SSI umgesetzt und können mit Live-Input getestet werden.

Schlagwörter:
Verarbeitung sozialer Signale, Erstellen online-fähiger Systeme, multimodales Framework

Abstract

In recent years, *Social Signal Processing* (SSP) has gradually emerged as a new research field in human-computer interaction aiming to build a new generation of "social computers", which will be perceived as more natural, efficacious and trustworthy. So far, most studies have been conducted offline and have been investigated social interaction under laboratory conditions. The problem of plain offline studies is that they tend to convey a too optimistic picture of what can actually be achieved since they avoid problems which occur only when a system is tested in the "open world". Hence, despite a good deal of work carried out in the field of SSP, we have not seen many online systems.

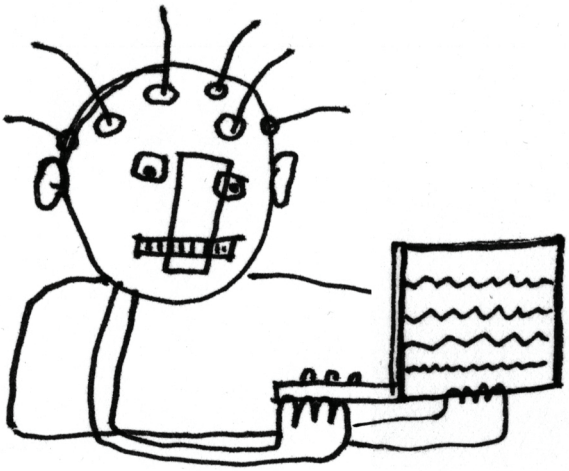
It is the goal of this thesis to encourage developers to put more effort into building online systems instead of confining their work to pure offline studies. Thus, this study examines the limitations of previous studies and proposes methodological and technical solutions, which pave the path towards a more realistic and application-oriented development. Today, most available SSP corpora contain audiovisual material composed of discrete and exaggerated samples. To ease data collection and enrich corpora with additional measurements, a mechanism will be proposed that allows for the synchronisation of audiovisual signals with other signals such as motion, eye gaze, and physiological feedback. To combine information from such diverse sources intelligent fusion strategies are needed. Unfortunately, conventional approaches have shown poor results in realistic scenarios. Thus, this study describes a novel fusion algorithm which allows for a better modelling of the temporal dependencies between modalities. Finally, it is common practice in offline studies to tweak the database to suit the classification process, for instance by removing parts with sparse interaction and non-prototypical behaviour. This can lead to suboptimal results if the learning process does not fit the final application. Therefore, a more application-related methodology will be proposed making it more likely that a system will perform satisfactory when transformed into an online approach.

In order to meet the increased implementation issues of developing real-time systems, tools must be provided that take as much work off the hands of the developers as possible. With this in mind, a new open-source framework called SOCIAL SIGNAL INTERPRETATION (SSI) will be introduced. It supports the complete process of the machine learning pipeline and provides a software architecture to accomplish complex processing pipelines from single, reusable units. It implements the discussed synchronisation and fusion mechanisms and systems developed with SSI can be immediately tested using live input.

Keywords:
Social signal processing, building online systems, multimodal framework

Danksagung

An erster Stelle möchte ich mich bei meiner Betreuerin Prof. Dr. Elisabeth André für die stets angenehme Zusammenarbeit und die hervorrangende Unterstützung bei der Erstellung der Arbeit bedanken. Auch geht mein Dank an meine beiden Zweitgutachter Prof. Dr. Björn Schuller und Prof. Dr. Bernhard Möller. Weiter bedanken möchte ich mich bei meinen Kollegen Florian Lingenfelser, Tobias Baur, Ionut Damian, Felix Kistler, Andreas Seiderer und Simon Flutura, die mich bei der Implementierung des SOCIAL SIGNAL INTERPRETATION Frameworks unterstützt haben. Ein ganz besonderer Dank gilt dabei meinen beiden Bürokollegen Florian Lingenfelser und Tobias Baur für die freundschaftliche Atmosphäre und die vielen interessanten Gespräche in unserem Büro und auf diversen Dienstreisen.



"Our technology, our machines, is part of our humanity.
We created them to extend ourselves,
and that is what is unique about human beings."

Ray Kurzweil

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Imagine the following scene: Someone approaches you on the street asking for the fastest way to the hospital. How will you react? Well, it depends...

Scenario 1 - The calm businessman: The person is a businessman in a tidy suit talking in a calm voice. You may assume he has a commercial interest and needs to keep an appointment. Yet, his calm voice tells you he is not in such a hurry. So you suggest a route that avoids off-road shortcuts because you do not want him to get dirty.

Scenario 2 - The old, confused lady: The person is an old lady with a rollator. In this case, you want to make sure the route is free of barriers even if this means she has to make a detour. However, since she seems to be confused and her overall condition is not too good you may offer to call someone to pick her up.

Scenario 3 - The desperate woman: The person is a woman all in tears. Maybe something bad has happened to her child? In any case, it feels like an emergency, so you may propose to call an ambulance or offer to lead her to the hospital in order not to waste any more time. Off-road shortcuts are welcomed.

Now, pick up your smart phone and ask the exact same question to Siri or another "intelligent" virtual agent. From a technical point of view, you will get a pretty solid answer. The software may suggest several alternate routes, include a map of the surrounding area, and give estimations of the time it will take to get there. In many respects, the answer will be more reliable and more detailed than that of a human. But it will be the *exact* same answer no matter whether the calm businessman, the old lady, or the desperate woman is asking. This is because Siri considers only the verbal part of the message. However, there is another level in human communication, which is just as important as the spoken message: *nonverbal communication* [207].

According to Watzlawick [340] a message can be decomposed into *what* is said and *how* it is said. In our example *what* encodes the query for a specific location, whereas *how* carries a range of additional information such as age and gender, attractiveness, body gestures, gaze motion or facial expressions. These *social cues* [97] can fundamentally change the interpretation of a message. For instance, they could tell us that the person who is asking for the way is in a hurry and therefore may prefer a quick instruction over several alternate routes. The ability to appreciate peoples' feelings, fears, and motivations to interact effectively with others is called *interpersonal intelligence* [120]. In addition to effective verbal communication, it requires the ability to correctly interpret signs of nonverbal communication - something computer engineers have paid little attention to until recently. This thesis tackles the question of how we can enrich the precise and extremely useful functions computers already offer with the human's ability to shape the meaning of a message through nonverbal messages.

1.1 Social Signal Processing

Only with the new millennium, researches began to recognise the great potential nonverbal communication bears for a more intuitive human-computer interaction (HCI), which is making use of natural and implicit user input instead of limiting itself to explicit keyboard and mouse commands [239]. Two milestones were particularly influential. In 1997, Rosalind Picard published her book *Affective Computing* [248] in which she declared that building intelligent machines means to "*give computers the ability to recognise, understand, even to have and express emotions*". About a decade later, Alex Pentland coined the term *Social Signal Processing* (SSP) [242] claiming that "*building socially aware systems [...] can provide a new dimension of communication support*". In the long run, such attempts aim to create a new generation of computers that allow for a completely natural and human-like interaction.

Nevertheless, although a good deal of work has been carried out in this area, this research has not yet translated into many applications. Why is this?

A review of the literature of the last decade gives a useful hint to answer this question. The probably most comprehensive source of information about Social Signal Processing is the *SSPNet Portal*[1], a European collaboration aimed at establishing a research community in SSP. The site was founded in August 2009 and offers an exhaustive and steadily growing bibliography including important SSP works and background to enter the field. Searching the listed publications for the term "real(-)time" within title, abstract, keywords, and content returns 135 paper, which is approximately one quarter of all papers that were available in October 2013[2]. A closer review

[1] `http://sspnet.eu`

[2] Retrieved on all papers since 2000 excluding papers related to synthesis. It should be noted that a paper was numbered as long as the term "real(-)time" occurred anywhere in the text, even if real-time processing was actually

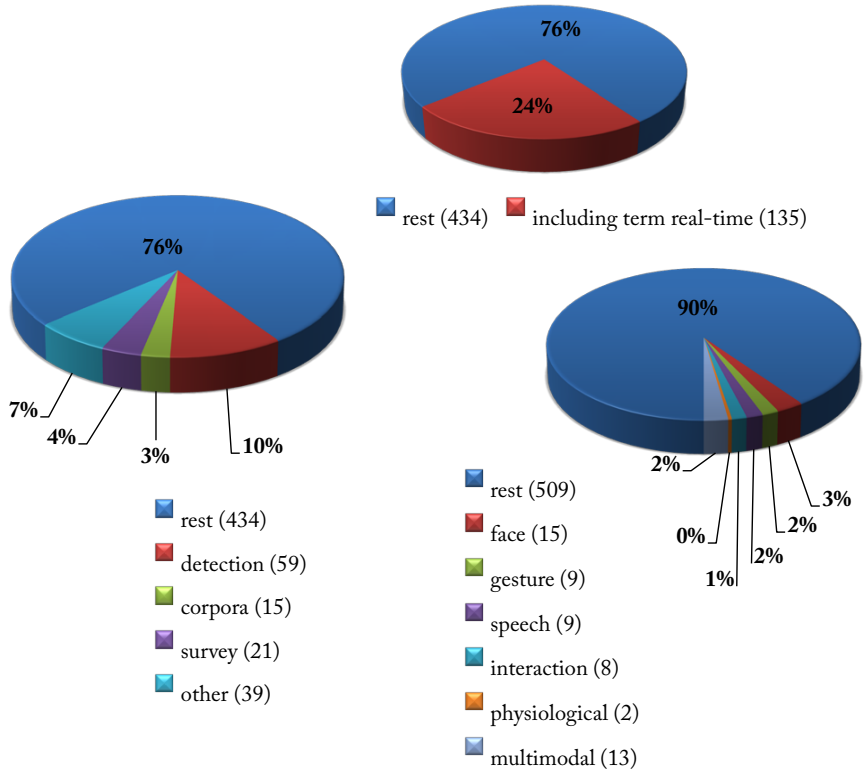


Figure 1.1: Number of papers listed on *SSPNet Portal* including the term "real-time" sorted by topic (retrieved in October 2013). The graph in the right lower corner identifies only ~2% as being related to multimodal detection.

of these papers revealed that ~10% of them deal with detection, while only ~2% address multimodal detection in particular (see Figure 1.1). This is indicative of the trend that papers explicitly addressing the problem of real-time detection are rather the exception than the rule, especially in case of multimodal input. Most SSP publications are offline studies.

One major problem of offline studies is that the reported accuracies themselves are not particularly conclusive. In fact, from the numbers reported in literature it appears that many problems are already solved. However, many studies make unrealistic assumptions, which are hard to meet in real-life environments; others are based on small databases, which are too specific to provide findings that could be generalised immediately. A way to make results more convincing is by providing a common data basis to serve as a benchmark. Recently, attempts have been made to host regular challenges in which researchers from around the world are invited to test methods on a common dataset using a standardised evaluation metric. A positively received effort is the *Paralinguistic Challenge*, which has become an annual event at the Conference of the International Speech Communication Association INTERSPEECH [290, 292, 294–297]. Every year, the organisers of the challenge provide several databases together with a baseline, which they arrived at within a state-of-the-art setting. This allows participants to compare their methods and estimate their usefulness.

not the main focus of the paper. Otherwise the amount of relevant papers were even less.

Although competitions allow for a better assessment of the reported recognition results, they still encourage researchers to conduct studies in an offline fashion. And in fact there are some good reasons to choose an offline system over an online system. Offline systems are generally easier to build, easier to maintain, easier to evaluate, and easier to publish. Approaching a problem in isolation is not a bad thing per se. But it remains a simplification which blanks out many of the real challenges. For instance, in offline studies it is common practice to remove noisy data or exclude parts with sparse interaction. This simplifies the classification process, but it avoids situations with which a system typically has to deal in a realistic environments.

1.2 Research Questions

It is my hope that this thesis will encourage developers to put more effort into building online systems instead of confining their work to pure offline studies. In order to achieve this goal this study provides a thorough review of academic literature on SSP during the past decade and works out major challenges that need to be tackled to build systems that detect nonverbal behaviour in real-life environments. Furthermore, solutions will be presented that facilitate taking the step from offline studies towards a more application-oriented methodology in SSP.

1. The progress from exploratory research towards real application relies heavily on the development of appropriate databases [80]. Hence it must be our objective to collect comprehensive training corpora stocked with sufficient variations of social behaviour to cover as many contingencies as possible. In particular it means turning away from what has been regarded as a classic database design, i.e. a collection of prototypical samples recorded by actors. This, however, requires more elaborate recording setups and also makes it more difficult to distill descriptions directly from the experimental setup. To this end, we will propose an appropriate database design and analyse what can be done to ease data collection, both in terms of recording and annotation.

2. Humans express nonverbal behaviour through a variety of channels such as facial expressions, expressive gesturing, or emotionally coloured speech. At times the information carried in other channels is redundant, but just as often it is complementary [356] or even contradictory [81]. This holds for realistic scenarios, in particular [17]. Hence, smart ways of fusing all available information are required to achieve a complete picture of the observed behaviour. Unfortunately, standard fusion algorithms developed on acted data generally show lower performance for realistic scenarios [78]. Therefore, we will identify the shortcomings of conventional fusion schemes and propose a novel fusion approach that allows for a better modelling of the temporal dependencies between modalities.

3. In offline studies, it is common practice to take a set of prerecorded files, balance the number of samples per class, remove parts with noisy or sparse interaction, extract and then normalise features. Finally, classification parameters are fine-tuned until a good configuration is found. The problem of such a methodology is that it says little about the applicability in real applications. In the following, we will analyse what it takes to turn an offline system into an online system. In particular, we will elaborate on the problems involved in an "open recording" setting, i.e. a setting where incoming data has to be processed as it comes, and suggest suitable strategies to cope with them.

4. Despite a fair number of offline tools, there is still considerably less support to accomplish real-time applications. The few online systems that are available today usually impose restrictions that can hardly be met outside the lab or are only able to recognise very exaggerated and prototypical user behaviour, which is rarely seen in real-life interaction. Moreover, they are often provided as black boxes which are limited to a certain purpose, either because they are distributed as closed source or because they lack a modular design making it difficult or impossible to adapt to a different context. To this end, we will introduce a novel framework called SOCIAL SIGNAL INTERPRETATION (SSI), an open-source software to record, analyse and fuse multimodal social signals in real-time.

1.3 Outline of the Thesis

The thesis is outlined as follows:

- **Chapter 2** summarises the basic theories and debates of a whole generation of psychologists on how to describe and classify the complex nature of nonverbal communication. It provides a description of the large repertoire of social cues together with their functions in human communication and ends with a discussion on emotion theory. The findings thus presented provide the theoretical background for the implementation of a machine that is able to recognise and understand nonverbal behaviour.

- **Chapter 3** outlines a general methodology of how to learn and interpret social signals with a machine. The important concepts of features and classification are demonstrated by means of vocal and facial affect recognition. The chapter closes with two submissions to the aforementioned *Paralinguistic Challenge*. One deals with the problem of removing parts of an utterance that are least relevant in a speaker trait recognition task. The other investigates the use of phoneme detection to predict laughter and fillers in human speech.

- **Chapter 4** reviews literature on SSP and works out challenges, which need more attention if we want to move towards more realistic and application-oriented systems. This

particularly concerns the need for large multimodal corpora, novel ways of fusing information from multiple channels, and a more application-related methodology. Each section is concluded by suggestions on how to tackle these challenges.

- **Chapter 5** introduces SSI and explains how the lessons learned in the previous chapters have been taken into account during the development of the framework. Crucial design aspects are highlighted, including synchronisation of multiple data streams, generic signal handling, and event-based communication. Standardised ways of storing and exchanging data are discussed as well. Finally, the important concept of a pipeline is presented, both in code and in XML.

- **Chapters 6 and 7** demonstrate SSI by means of a concrete recognition pipeline which is gradually advanced to a complex multimodal recognition system. Special attention is paid to the modular design of SSI, which allows developers to quickly adapt consisting pipelines to their needs. Thus, the final system handles typical problems an online system has to cope with, such as continuous recognition, missing data, and incremental detection.

Chapter 2

Nonverbal Communication

After returning from Southeast Asia, a friend of mine told me about a young man she had met on her trip. Since he had no English skills and both of them did not speak the mother tongue of the other, it was impossible for them to exchange even basic verbal messages. Apart from some numbers they were writing on a mobile phone, body language and sounds were all they had to communicate. Still they ended up spending the whole evening together and even arranged to see again the next day at a flea market. Back in Europe my friend described the young man as one of the friendliest people she had ever met. How is that possible when they were actually not able talk to each other?

It is possible because human communication is more than words and can be exchanged through various other channels, such as body gestures, paralinguistics and facial expressions to name only a few. These channels build the foundation of what we call nonverbal behaviour and they play an important but easily overlooked role in human-human interaction. When we tend to neglect their importance it is only because large parts of nonverbal communication happen outside our awareness. Only when circumstances force us to abandon language and stick to nonverbal ways of communication, as in case of my friend, we suddenly become aware of their utility and power. This, however, does not mean that nonverbal behaviour is less important in cases where we are able to communicate through speech. In fact, as we will see in the following, nonverbal behaviour plays a fundamental role in human communication and our lives would be a different if we cut it off.

2.1 *What* versus *How*

> *„One cannot not communicate"*

.. is the first of five axioms defined by psychologist and philosopher Paul Watzlawick in his theory on communication. It means that any human behaviour is a kind of communication. Even

if we are silent and inactive we convey a message, at least as long as we are in the presence of others [340]. For instance, if we see a man in the waiting room who is constantly starring on the floor we would understand this as a sign to leave him alone. On the other hand, if he is turning his head towards us showing a friendly look, we may take this as an invitation to start a chat. Of course, his reaction also depends on our own behaviour, whether we are open to a conversation and to what extent we project this to our environment. In any case, the decision to initiate a conversation will primarily be made at a nonverbal level. That is because human communication is more than words and can be exchanged through a various number of channels and their combinations [340].

Even if we exchange greetings with the man in the waiting room, it will not be the meaning of the words alone, but rather intonation and bodily behaviour, which allows us to estimate if the other is open to a conversation or not. This highlights an important aspect present in all communication, Watzlawick calls the *content* and the *relationship*. He illustrates it by means of a computer that needs both: data, which is the information, and a program, which is the information about this information. None of the two inputs alone is sufficient for its function. Transferred to our previous example we cannot even exchange a simple greeting (*content*) without revealing *how* we want the message to be understood (*relationship*) and usually it is the *how* which is transferred through nonverbal cues. The tone of our voice, our facial expressions, our gaze motion - intended or unconscious - encode whether we like or dislike a person, want to get to know him or her better, start a conversion or avoid further contact. How the implicit cues attached to a message are received by the other person and whether they are correctly understood, however, remains open. For instance, the man in the waiting room could be introverted and for that reason refrain from directly looking at us, but we may falsely interpret his behaviour as repelling. Of course, it also depends on the context where the interaction takes place. For instance, when someone we do not know approaches us in an environment less qualified to start a conversation with a stranger (e.g. while walking through a populated street), we may assume we are not intended or taken for someone else, and this time it could be us who is reacting repelling.

To sum up, we can retain that humans exchange messages on two levels. At a verbal level, mainly used to exchange explicit messages, i.e. *what* is said, and at a nonverbal level, usually used to encode implicit messages expressing *how* we mean or receive something. When talking we always send out messages of both kinds, and often verbal and nonverbal parts complement each other. However, they may also express contradictory information and in this case it is the nonverbal part that leaks information how we really think and feel about each other. During moments where we are not talking ourselves, we still make use nonverbal signs, e.g. to express if we agree with something said by another person. Hence, it is often the nonverbal hints that really matter, either because they reveal some hidden aspects about our intentions, or simply because they are the only source of information available. Although it is hard to verify what percentage of our communication is nonverbal, we can assume that it forms the bulk of human

communication[1].

In contrast to verbal communication, which is based on a single channel, namely speech, nonverbal communication has various sources. Drawing a complete picture of the expressed behaviour requires the combination of information from several if not all sources, where each bit of information defines a behavioural cue. Hence, in the later course of this thesis we will not only discuss the problem of detecting these cues, but also think about ways of fusing them. But first we want to answer the question why people draw on nonverbal communication, when they already have such a powerful communication tool as the human speech, and how nonverbal signs are expressed?

2.2 Functions

In contrast to speech, which at least at this form of complexity is a unique achievement of mankind, nonverbal signs can be found everywhere in the animal world. This was already perceived by Charles Darwin, who was the first studying nonverbal communication in his 1872 published book *The Expression of the Emotions in Man and Animals*. Hence, nonverbal behaviour has survived as part of our behavioural repertoire having a much longer history than verbal communication. And like speech some of it is generated on purpose, e.g. a pointing gesture or a sequence of head nods to give feedback without interrupting a talking person. However, in large parts it is leaking unconscious and hard to control, e.g. opening of the eyes in reaction to a surprising event. When behavioural signs are unwillingly emitted, we say that we are *giving off* information, either because we do not intend to give an information, or because we just do not know it.

Often we use nonverbal messages to accompany speech. We point at the object of discussion to give additional hints (repeating) or to make it easier for the listener to follow our explanations (complementing). Sometimes because we miss the name for it (substituting). In other situation we send opposing messages, e.g. to indicate the ironic meaning of what we say (conflicting). Touching someone's arm can signal that we like to interrupt (regulating). And we amplify or tone down aspects of verbal messages, e.g. by shaking a fist while shouting at someone (accenting/moderating). Nonetheless, there are functions of nonverbal behaviour.

Imagine you are at a party and your friend introduces you to someone you have not met before. As usual you start by talking about informal topics, current sports events, what is in the movies,

[1]Derived from experiments dealing with communications of feelings and attitudes (i.e. , like vs. dislike) a study by Albert Mehrabian suggests that the judgement of another person depends only to 7% on the words, while about 93% percent is communicated nonverbally. Although it is important to note that this rule holds only under such specific conditions [207]. More recent research assumes that between 60% and 70% of all meaning is derived from body language [102]

the weather, and such. After a few minutes of small talk you have not touched any personal issues. Still, you have made your judgements whether you are interested in the other person or not. And the decision to continue the conversion will depend on this first estimate[2]. In fact, many of the judgements we make about others in everyday life are based on what Ambady *et al.* denote as *thin slices of expressive behaviour* [5]. And their meta-analysis proves that we are not only fast, but also surprisingly accurate. Predictions based on observations under 1/2 minute in length do not significantly differ in accuracy compared to predictions made on 4-5 minutes observations or longer.

In sum our ability to emit and perceive nonverbal cues allows us to intuitively make fast and reliable decisions in situations where speech is not adequate. It helps avoiding embarrassing situations, as nonverbal communication is considered more polite than verbal [10]. It saves us from being cheated as it helps revealing lies or deceptions [93], but also works as *social glue* [184], e.g. we can gain trust from social partners by giving insights in our feelings and emotions, or enhance affiliation and liking by adopting similar nonverbal behaviour (*chameleon effect*). In fact, it is argued that the ability to express and recognise social signals is indispensable for success in life [127].

Picking out a certain nonverbal behaviour we may wonder why it manifests itself in this particular form. For instance, why do we wrinkle the nose when we are disgusted or bare our teeth when enraged. A possible solution has been proposed by ethologists saying that nonverbal behaviour primarily had a specific function directly linked to survival; wrinkling the nose reduced the inhalation of foul odors and baring our teeth is a remnant from times when biting was used as a weapon [179]. But over time the original function got lost and instead was provided with a communicative value [313]. However, Krauss [179] warns not to use the terms nonverbal behaviour and nonverbal communication interchangeably, as not all nonverbal behaviour must necessarily has a communicative purpose. He distinguishes between *inter*personal and *intra*personal functions of nonverbal behaviour. Only *inter*personal functions actually convey information to others, while *intra*personal functions do not serve a communicative purpose. Periodic fluctuations of a speaker's gaze, for instance, could simply be a consequence of two complex tasks a speaker is involved in. First, he or she has to formulate the spoken content and turning away from the dialog partner helps to reduce the visual load. Afterwards, feedback is required and now the gaze is redirected toward the listener.

[2]Of course, first estimates can be wrong and for more than once we may have repelled someone for no good reason. But unless you are person who is extremely unbiased you may stick to your judgement and it will take much longer to change your mind than it took you to adopt this attitude in the first place.

2.3 Social Cues

During social communication we make use of a large repertoire of nonverbal cues. Almost all parts of the body are involved in their generation, ranging from the body as a whole, e.g. when taking a certain position towards another person, to small parts such as the eyes, e.g. when giving someone a wink. A single cue may last from a few millisecond (a blink) to several minutes (sitting) and is defined by how it became part of the person's repertoire (*origin*), the circumstances of its use (*usage*) and the information it conveys (*coding*) (see [97] for a detailed discussion).

The *usage* of certain behaviour may be restricted to a custom environment, e.g. our workplace or to family members. As mentioned earlier, it may occur in relation with speech or as a feedback mechanism. It differs in the level of awareness and whether it was intented as a communicative act. And if the information associated with it has a idiosyncratic or shared meaning. However, there are also meaningless actions, e.g. when changing position for comfort.

The *origin* of an action can be a reflex, e.g. when we turn away our head to avoid a hit. It can be acquired as a consequence of a species-constant experience, e.g. using our hands to place food in the mouth. Or it can be learned in culture-specific experience, either as a result of a particular activity such as driving, or as part of social interaction. This may happen explicitly, e.g. following the instructions of a teacher, or unconsciously by imitating another person's behaviour.

The code of a message can be *extrinsic* or *intrinsic*. The difference can be explained by means of the sign language alphabet, where some signs resemble visually the written version of the according letter (*intrinsic*), while others look nothing like their written equivalent (*extrinsic*).

Based on the particulars of usage, origin and coding, Ekman and Friesen [97] distinguish five categories of nonverbal behaviour: *emblems*, grouping nonverbal acts with a direct verbal translation normally employed intentionally, e.g. winking or thumbs up; *illustrators*, which are movements directly tied to speech and slightly less controlled, e.g. finger pointing or raised eyebrows; *displays of affect*, primarily expressed through the face, usually in awareness but often without deliberate intention to communicate; *regulators*, which are acts to mediate between conversational partners most of the time performed subconsciously, e.g. head nods and eye contact; and *adaptors*, which are movements usually first learned in childhood and later unwillingly emitted by habit in reaction to certain stimuli, e.g. wiping around the corners of the eye in the sensation of grief, originally to clear away tears, but still used by adults with no tears present.

In the following we will list the most important behavioural cues together with their function in social communication.

2.3.1 Appearance

Appearance includes properties of the human body such as height and attractiveness, but also artificial characteristics, such as jewellery and clothes. The influence of these variables is closely related to the *halo effect* found by Edward Thorndike in the 1920s [312]. Thorndike asked officials to evaluate their soldiers and found a too high correlation of intelligence rating with the ratings for physical attributes, leadership skills, and personal qualities. Generally the *halo effect* describes the phenomenon of carrying-over one judgment to another. Dion *et al.* [77] proved that more attractive people are credited with more socially desirable personality traits and to have a happier and more successful life. Similar stereotypes apply to vocal attractiveness, where senders with more attractive voices are rated more favourably than senders with less attractive voices [364].

2.3.2 Facial Expressions

Facial expressions are contractions of the muscles of the face and our primary channel to express emotions, attitudes, and moods. Apart from that people use it to provide nonverbal feedback during conversations and to reflect their interpersonal attitudes [176]. While we are at least to some extent able to control our facial expressions, many unconscious facial expressions exist which leak our true feelings even if we do not want them to be observed by others. These involuntary facial expressions are called microexpressions as they are very brief in duration, usually less than half a second [91]. The generation of facial expressions involves movements of the lip, cheek, brow muscles and other parts of the face, often in combination.

During interaction with others we use the face to signal opening and closing of a conversation, complement or qualify verbal and/or nonverbal responses, and to replace speech [176]. Smiling, for instance, while usually thought of as an emotional cue, can also express our desire to start or end a conversation. During a conversation smiles are often used as a signal of attentiveness and involvement [34]. Sometimes, we use facial expressions as what Ekman and Friesen denote as *facial emblems* [99, 136]. That is, we show an emotional expression without actually being in the emotion. For instance, we may wrinkle the nose to share our disgust for a situation someone is telling us about.

The repertoire of facial expressions our face is capable of making is tremendous. During his study of the human face, Ekman discovered more than 10,000 different configurations [90]. Most studies, however, are restricted to a small number of *basic* emotions, although there is no general agreement which emotions should be considered as *basic* [307] (see Section 2.4). A prototypical expression of surprise, for instance, would go along with raised eyebrows, horizontal wrinkles across the forehead, open eyelids, where the upper lid is raised and the lower lid is drawn down, and dropped jaw so that the lips and teeth are parted, with no tension around the mouth [99].

In everyday life, however, we do not always portray *pure* emotions, but a blending of multiple emotional states, so called *affect blends* [176]. For instance, at the moment when we notice that our purse was stolen we may feel both, anger towards the thief and grief over the loss.

During the first half of the 20th century it was generally thought that facial expressions are entirely learned and culture-specific. Paul Ekman's studies with people of a very isolated population from New Guinea changed this view [98]. From a set of photos showing basic facial expressions of Western people, participants had to select the photo which in their opinion matched best with a story they had heard. Except for fear and surprise, which were constantly mixed up, it turned out that the inhabitants of these isolated mountain villages had no problem to associate a story with its corresponding expression, although they had never seen a Western movie or lived in a Western country. Likewise, when showing photographs taken of New Guineans while showing how they would react in certain situations, U. S. citizen were able to guess the current scenario [98]. Expressions Ekman found to be universal include anger, fear, sadness, enjoyment, disgust, and surprise [95].

In 1978, Ekman and Friesen published the Facial Action Coding System (FACS)[3], a system to taxonomise human facial expressions [92]. It is based on a set of 64 Action Units (AU) which allow human coders to describe nearly any possible facial configuration. AUs are a contraction or relaxation of one or more muscles. This allows, for instance, to distinguish a faked smile generated by a raise of the lip corners (*zygomaticus major*, AU 12) from a sincere smile, a so called Duchenne smile, which also involves a contraction of the muscles responsible for raising the cheeks (*orbicularis oculi*, AU 6) [94]. FACSAID (Facial Action Coding System Affect Interpretation Dictionary)[4] is a variation which considers only facial actions relevant for the expression of emotion.

While we obviously use the face to express our emotions, we should bear in mind that our facial expressions must normally not be taken as a one-to-one mapping of our internal feelings. In fact, they largely depend on the social circumstances and social goals we are involved in [176]. For instance, people that are in a happy mood do not necessarily smile as long as they are not in presence of others [113, 180]. And facial reactions may be stronger in front of friends as opposed to strangers [330]. Some researchers even hold the extreme view that facial expressions are always enacted for social purposes [118]. As is so often the case, the truth probably lies somewhere in between as we display both, spontaneous and deliberate expressions.

[3] `http://face-and-emotion.com/dataface/facs/description.jsp`
[4] `http://face-and-emotion.com/dataface/facsaid/description.jsp`

2.3.3 Gaze Behaviour

Within the human face, movements of the eyes are considered as another important carrier of nonverbal behaviour. Interestingly enough, the widely exposed white sclera (white of the eye) surrounding the darker coloured iris is one of the characteristic of humans not found in other primate species. Obviously designed for communication it makes it easy for others to discern the direction of gaze [177]. Although we are not always aware of it, we adopt a number of eye-related norms during our life. For instance, we learn that it is inappropriate to stare at strangers or to look at certain body parts in public [176]. And during conversations we perceive constant eye contact as unpleasant.

Kendon [168] analysed gaze direction in films of two-person conversations. He noted a monitoring function of gaze. Looking at each other provides information about the other person's behaviour. Apart from that he found a regulatory function in terms of turn-taking. When starting a sentence speakers tended to look away from the listener, but as soon as a turn change was due they directed their gaze back to the listener. Finally, speech flow may be reflected as well. A speaker looks more in the direction of listeners when talking fluently and less if he is hesitating. Argyle and Ingham measured the amount of gazing in two-person conversations and noticed that people gaze nearly twice as much while listening (75%) as while talking (41%) [9].

Apart from a regulatory and monitoring function, the eye area can also provide information about our emotional state. Tears, for instance, are a clear sign of emotional arousal, although out of the context it remains unclear whether they reflect grief, physical pain, joy, or some other emotion [176]. Again, studies by Ekman and Friesen [99] reveal connections between configurations of the eye and basic emotions. In case of fear, for instance, they report the white of the eye begin exposed, while the upper eyelid is raised and the lower eyelid is tensed and drawn up.

After all, gazing also gives off hints about the nature of the relationship between two people. How much we look at a person while he or she is speaking may depend on the status of the person as well as our own status. High status males, for instance, receive more visual attention than low or equal status males [106]. However, more or less eye contact can also serve as a measure of how likeable we find our counterpart [207].

2.3.4 Vocal Cues

Speech offers humans a medium to communicate their thoughts, intentions, memories, and knowledge. In particular, it allows us to share feelings with others, and not only by words, but also by how something is articulated. This is the paralinguistic part of a spoken message. A model, which tries to explain the process of vocal communication of emotion as a whole, has been suggested by Scherer [276]. It is based on Brunswik's lens model of perception [35] and distinguishes three events: *encoding*, *transmission* and *decoding* (see Figure 2.1).

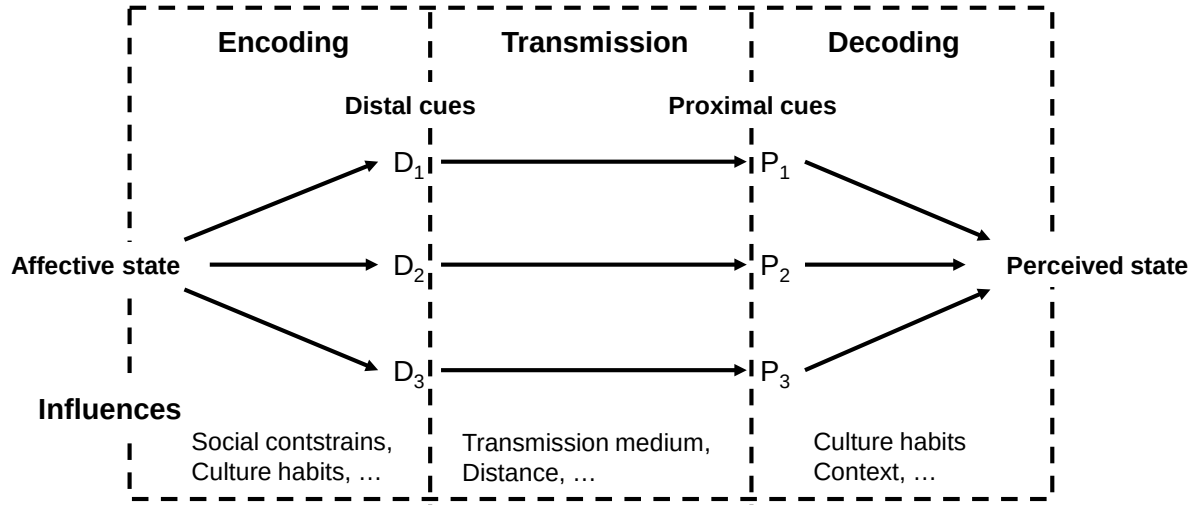


Figure 2.1: Simplified Brunswikian lens model for vocal communication of emotion, according to [276]

Starting with encoding, the speakers' momentary affective state causes changes of respiration, phonation, and articulation, which are acoustically reflected in the speech signal as *distal cues*. Following the transmission phase, the observer perceives these cues as so called *proximal cues*, from which he can deduce the speaker's emotional state (decoding). According to Scherer's point of view, the proximal representation of the distal cues should be defined as the outcome of the auditory cortex, i.e. an already interpreted version of the input signal. Hence, the proximal cue for the fundamental frequency would be the perceived pitch.

During the process, several interferences may occur, which influence the modification and perception of emotional speech. For example, social constraints can keep the speaker from showing what he really feels, and the context under which communication takes place can serve as an additional hint for the observer to correctly interpret the message. Finally, the transmission medium, whether it is, for instance, directly through the air or via telephone, has an impact on the perceived voice quality.

Linking physiological measurements with the acoustic assessment of emotional speech has been the focus of Johnstone *et al.* [160]. In their experiments, the authors found significant emotion effects on the closing time of the glottis. For high arousal emotions with high pitch and energy, they measured a faster closing of the glottis (proportional to the fundamental period), which is a sign of increased vocal effort and/or laryngeal muscular tension as in pressed voice [111, 174]. Further, they found lowest pitch values for the emotions bored and depressed, and highest for happy and anxious speech, which they trace back to the increased muscle tension in high arousal emotions. They also assume that the limited pitch range they found for tense, irritated and anxious speech reflects a general tenseness in the laryngeal musculature that limits adjustment of vocal cord tension and larynx position. Other experiments revealed a correlation between

respiration and speech: in certain situations participants reported as stressful, longer respiratory cycle and decreased respiratory depth appeared with higher pitch range. In contrast, during contented or calm situations speech production was more relaxed and thus respiratory depth and rate of articulation was normal.

Using an electromagnetic articulography system (EMA) movement of tongue tip, jaw and lower lip during emotional speech has been tracked in [103, 104, 190]. Findings reveal that emotional speech articulation exhibits more peripheral or advanced tongue positioning, especially for sad emotion. Also, for angry speech larger jaw opening and greater tongue tip velocity is reported. As a surprising outcome, happy speech articulation was found to be more or less similar to neutral speech articulation, except for a wider and higher pitch modulation. It is pointed out that the intensity of emotional reactions varies for different vowels.

Recently, Magnetic Resonance Imaging systems (MRI) have been used to obtain images of the vocal tract during emotional speech production. In contrast to EMA systems, observations based on MRI are not restricted to these parts of the vocal tract which are accessible from outside. Using a real-time MRI system the authors of [191] report most active tongue-tip movement during angry speech, which is conform with previous findings by EMA systems. However, the study also proves that angry speech is most active not only in the mouth cavity but also in the pharyngeal region. Tracking of the vocal chords disclosed that the overall active vocal tract length is shorter in happy speech. Also, for angry speech a wider vocal tract shaping was measured. By collecting additional data from more subjects the authors hope to find some common characteristics of emotional articulation across speakers.

2.3.5 Postures and Body Movement

Postures and body movement are argued to give off most reliable cues about the actual attitudes of a person towards social situations [261]. That is because they are typically performed unconsciously. A positive attitude towards others is expressed by increased degrees of *immediacy*, which Mehrabian defines as the *directness and intensity of interaction between two entities* [208, 209]. *Immediacy* is directly linked to greater degrees of touching, forward lean, eye contact, and body orientation. As a second primary dimension of posture Mehrabian defines *relaxation* and indicates e.g. arm and leg asymmetry as as important determiners. He notes, however, that effects of postures need to be specified depending on the social status of communicator and addressee.

Another frequently observable phenomenon, although it almost always goes off completely unconscious, is posture sharing. Posture sharing describes the convergence of both the body positions and the movement qualities between dialogue partners. It is also known as *chameleon effect* and not necessarily limited to postures, but also applies to mannerisms, facial expressions,

and other behaviours of one's interaction partners [49]. When observed it can be taken as a positive indicator for rapport, liking and affiliation [184]. Taking of the other's role in the course of interaction signals agreement and expression of a shared perspective [223], and it helps fostering relationships with others [184].

Scheflen [271] defines three types of postural configurations. One concerns the mentioned *chameleon effect* and distinguishes *congruent* from *non-congruent* behaviour. The other classify postures into *inclusive* vs. *non-inclusive* and *vis-a-vis* vs. *parallel*. For instance, people forming a circle express an *inclusive* behaviour towards each other, while everyone outside the group is excluded. However, even within a group one can show signs of *non-inclusion*, e.g. by crossing ones arms or slightly turning away from the others. While talking to someone we can either stand *vis-a-vis*, which allows us to monitor the other and shows increased engagement. Or we can take a *parallel* orientation and focus attention to some external object.

In terms of affect display, posture ist sometimes argued to contribute less than other channels. In fact, research on emotion recognition has been mainly focused on face and voice [319]. Still, there are some studies suggesting that postures provide visual cues to emotions which go beyond simple emblems, such as a raised fist for anger. Regarding anatomical features angular postures rather convey threat, while rounded postures are perceived as warm and friendly [12]. When presenting computer-generated postures in a forced-decision task, where participants could assign one of six basic emotion labels (anger, disgust, fear, happiness, sadness, surprise), Coulson [56] found 90 percent agreement for some of the postures. For anger, sadness and happiness he even reported rates comparable to static facial expressions. On the other hand, there was e.g. low agreement on disgust. In another body expression-matching task, Van den Stock *et al.* [319] found fear and anger more poorly recognised than happiness and sadness. In a second experiment, when participants were presented compound images of faces on bodies, they were able to show that facial expressions were more accurately recognised if in line with the expression of the whole body. Moreover, the latter had more influence when the facial expressions were less extreme. Finally, a third experiment was conducted to investigate the influence of body language on recognition of voice prosody. Again, a bias towards the simultaneously perceived whole-body expression was found. Atkinson *et al.* [14] demonstrated that both, the kinematics of body gestures and the form, contribute emotion perception. Influence of kinematics was investigated by Crane and Gross[63]. They used motion capture to record walkers in two negative (anger and sad) and two positive (content and joy) emotions, and in neutral state. Emotions were elicited with aid of autobiographical memories and verified via self-report and by two observer groups. In the end only those trials where included in the analysis, where the target emotion was both felt and recognised. Within these trials, fastest speeds occurred in anger and joy trials, while gait speed was significantly slower in sad trials. Both posture and limb motions also changed with emotion. Perception of affect from arm movements was examined in [254].

2.3.6 Hand and Arm Gestures

Hand and arm gestures are movements performed with hands and arms (not all hands and arms movements are gestures, though). In general, we can distinguish three types of gestures. Those conveying a specific meaning, hence *emblems* [97]; examples are the thumbs-up or victory sign. Gestures related to speech, also called *illustrators* [97] or *conversational gesture* [179]. Unlike *emblems*, they do not occur in absence of speech and their temporal occurrence is well aligned to the spoken content; examples would be the rising of the forefinger to stress a certain statement or pointing at an object of discussion. Finally, there are gestures referred to as *adaptors* [97] which are performed completely unconsciously and are not perceived as communicatively intended. Still they reveal feelings or attitudes toward other people [244]; an example would be permanent scratching or rubbing as a sign of tension.

Regarding the communicative value of conversional gestures surprisingly little consensus is found in literature. A pro-communicative argument could be that face-to-face interaction is accompanied by more gesturing in comparison to interaction where speaker and listener are separated [264]. But whether a higher amount of gesturing is actually communicatively intended or simply due to the presence of others is not obvious [179]. Indeed, some studies put the communicative value of gestures in question. It seems that humans exchange information over the telephone just as effectively as in face-to-face situations [342]. Feyereisen *et al.* [114] conducted an experiment, where they showed subjects videotaped gestures together with three possible interpretations. When trying to select the correct response, subjects chose about as often the *implausible response* as they decided in favour of the *correct response*. Most of the time, however, they selected the *plausible response*, which was the meaning selected most frequently by an independent group of judges.

Krauss *et al.* [179] suggest that the main communicative value of gestures may not lie in the transmission of semantic information but reveal information of the speaker's internal state and attitudes toward the addressee, just like paralinguistic information does in speech. In terms of affect display, it is rather the *quality* of a gesture than its semantic meaning [26]. Castellano *et al.* [46] used quantity of motion and contraction index of the body as well as velocity, acceleration and fluidity of the hand to distinguish basic emotions. Recognition results were clearly above chance level and similar to those reported for speech and face analysed in parallel.

Apart from an *interpersonal* function, Krauss *et al.* [179] attribute gestures an important role in the speech generation process. In fact, there is evidence that preventing gesturing on speech actually leads to a decrease in the vividness of imagery during conversations [265]. Krauss *et al.* [179] assume that lexical movements are triggered by representations in short term memory that come to be expressed in speech. Again based on experiments, where speakers were prevented from using gestures, Rauscher *et al.* [258] found that subjects had difficulty of lexical access for speech with spatial content.

A problem that generally hampers the systematic taxonomy of body postures and hand gestures is the (theoretically) unlimited vocabulary and that perception may depend on the observer's viewpoint [66]. On these grounds there is no standardised coding system as FACS is for the face, yet. However, there is effort to establish such a system. Recently, Deal *et al.* [65] have developed a new method, the Body Action and Posture (BAP) coding system. BAP allows the description of a body movement in respect to three dimensions: involved body parts (anatomical level), direction and orientation of the movement (form level), and its communicative and self-regulatory functions. The Emotional Intelligence Academy (www.emotionintell.com) in collaboration with Paul Ekman have announced a comprehensive framework similar to FACS but applied to the body.

2.4 Emotion Theory

As we have just seen, nonverbal behaviour plays a crucial role in the expression of emotion. But what do we actually mean with *expression of emotion* and what tools are we offered by physiologists to formalise the phenomenon *emotion*? During our discourse we will use the terms *emotion*, *affect* and *feeling* interchangeably, as it is usually the case in everyday language. Although it should be noted that psychologists indeed make distinctions. Unfortunately – and as we will see soon this will also be the case for other concepts related to emotion – no definite definition exists. According to Damásio *affect* serves as the overall term, while *emotions* are taken to be more physiological describing what can be observed from outside, in contrast to *feelings* that are taken to be more psychological referring to an internal state [68]. As a general definition we can note that emotions (or affect or feelings) are brief and distinctive episodes which arise as a response to particular stimuli. They can last from some seconds to a couple of minutes, but not for hours or days which differs them from *moods*.

2.4.1 Functions and Effects

First, we may ask what functions emotions play in our life. One can see emotions as a sort of behavioural programme, which allows us to choose an (hopefully) appropriate reaction as response to a stimuli. And this means not only to decide on a certain kind of action but also at which intensity and duration it is performed. In contrast to conscious reasoning emotional reactions happen unconsciously, which allows us to make decisions faster. Obviously this is useful whenever immediate actions are necessary, e.g. in presence of danger, when the feeling of fear may unlock additional physical power[5].

[5]To some people it may also have a paralysing effect, which is why me must not assume that emotions always let us do the right things. Especially in our modern society, where threats are rather rarely posed by predators.

However, emotions are not only important in situations where we fear for our lives. The Nobel laureate Herb Simon was one of the first researchers to emphasise the influence of emotion problem solving [301]. During a fictive car purchase, Isen *et al.* proved that subjects put in a good mood took faster decisions as they were able to process the provided information more efficient [151]. Some years later he showed in a similar experiment that a positive mood also has a positive effect on creative thinking [150]. Yuen and Lee came to the conclusion that people in induced depressive mood have a lower willingness to take risk than people in neutral and in positive mood [354]. Generally, we can state that positive feelings improve our ability to solve problems and that too little emotion impairs decision making just as too much emotion does [248].

2.4.2 Models of Emotion

Reviewing emotion related literature, however, brings to light how many different terms researchers have used in order to describe distinctive emotional states. Some of them occur with higher frequency, such as *primary emotions* [250], *basic emotions* [307] or *full-blown emotions* [275]. No general accepted theory exists. A main obstacle is certainly the diversity of emotional states and that we often lack the words to describe them adequately enough. Also, emotions can occur in many different nuances and are rarely pure but often a blending of two or more. Accordingly, it has put to discussion whether emotions should be modelled as discrete entities or as part of a continuum [220].

Discrete Entities

In the case of discrete entities, one has to agree on a number of categories drawn from everyday language. Unfortunately, due to the quantity of emotion-related words that exists in language, disparity in this point could hardly be greater. In his "Semantic Atlas of Emotional Concepts" Averal has listed 558 words from English language which he found to be connected to emotion [15]. Although the number of words which could actually be used for describing emotion is smaller, Cowie *et al.* [80] claim 60 categories as a lower bound that should be considered if one wants to cover the full range of emotions in everyday life. A number that is hardly tractable wherefore an organisation into larger classes is needed.

For this purpose, Plutchick defines a small number of primary emotions (e. g. anger, fear, sadness, $\cdots$), which he believes can be blended together to form the full spectrum of human emotional experience [250]. An idea that is also referred to as palette theory [273], due to its similarity to the mixing of a set of primary colours to form combined colours. A similar concept is that of basic emotions maintained by Ekman [95]. He suggests considering only those emotional states which are primary, innate, and universal in all human beings. From investigations on the Fore

tribesmen of Papua New Guinea, he educed that even members of an isolated Stone Age culture could reliably identify some expressions of emotion. Thus, those emotions must be inborn. However, other theorists suggest different sets of basic emotions ranging from only two states up to nine or more [11, 119, 154].

According to Cowie *et al.* [80] the use of a small number of discrete emotion entities need to be treated with caution, since each category actually represents a family of related states, although conceptual closeness must not necessarily result in similar speech. As an example they cite pride, contentment, and zest, which could reasonably be gathered into one super ordinate category, but are likely to cause different vocal expressions. However, the authors still consider lists of emotion categories as significant descriptive tools. In [58] they appoint 16 key emotions, which are considered to be sufficient to describe emotions in everyday life, as proven by a representative survey.

Dimensional Representation

Alternatively, emotions can be represented as coordinates in a higher dimensional space. Theory based on this approach goes back to Wundt [352], who classifies emotions according to the following bipolar dimensions: *pleasure and unpleasure*, *strain and relaxation*, and *excitement and calm*. By spanning an emotional space with one axis for each variable pair, emotions can be evaluated according to the quantity of pleasure, strain and relaxation they contain or their antonyms. In a similar formulation Schlosberg has proposed valence and arousal as alternative dimensions [282], which build the basis for the popular *activation-evaluation model* described in [59].

An advantage of multi-dimensional models is their capability to express less intense and blended emotions. And although they are more tractable than words, they can still be translated into and out of verbal descriptions [61]. On the other hand, mapping to a small dimensional space can lead to a distorted picture about how emotional states are related to each other. For instance, an unwanted side-effect of the activation-evaluation space is that fear and anger appear to lie close together. A better distinction can be achieved by adding additional dimensions, for instance, *control*, as proposed in [231]. However, once additional dimensions are added to achieve a better description of a few additional states, it is difficult to know where to stop [61].

Appraisal

Appraisal-based theory [121, 275] is based on the idea that emotions are the result of our evaluation (appraisal) of a specific situation. In other words, emotions are individual reactions driven by our needs and concerns extracted from the evaluation of our own internal state and the state of

the outside world. Hence, emotions are neither reduced to a limited set of discrete categories, nor represented as a point in a continuous space spanned by few basic dimensions. They are rather described through a set of stimulus evaluation checks, including the novelty, coping potential, or compatibility with standards [133]. In particular, this model takes into account that individuals may respond differently to the same event.

Which of the models – categorical, dimensional or appraisal-based – should finally be preferred depends on the given task. Since strong emotion leaves people speechless or incoherent, as stated in [80], people who talk are usually in moderate and mixed emotional states. Thus, for speech emotion recognition a dimensional model might be more adequate since human dialogues require the machine to recognise mainly mild, not extreme emotional states. For practical reasons, however, categorical descriptions are still the more commonly used representation.

Chapter 3

Affective and Socially Aware Computing

In the previous chapter we have discussed the important role nonverbal behaviour plays in human communication. In particular, we have seen that nonverbal behaviour is linked to affective and social cues transmitted through body, face and speech [97, 207, 275]. Since most of these signs are passed on non-intrusively, it should be possible for a machine to intercept and interpret them just as humans do among themselves. In the remaining course of this chapter we will discuss potential benefits social computing will offer and think of potential applications. We will also examine the general steps that are necessary to detect nonverbal cues in an automatic manner. Finally, two paralinguistic studies are presented which alter the basic concept in some ways.

3.1 Human(like)-Computer Interaction

Some researchers have referred to computers as "socially ignorant" [243, 321]. But, we may just as well turn the tables and say that humans are "socially addicted". Indeed we are! From birth on we are involved in social interactions and until our dying day they will make up a very large part of our life. In fact, we are so much into our role of being a social animal that we cannot suppress our social skills when interacting with a machine. We realise this the very moment we find ourself shouting at a device that is not working the way we expect it to work. In fact, people regard computers as social agents [237] towards which they behave as they would in any social situation [221]. And from this point of view current computer systems are indeed socially ignorant lacking even the most basic abilities to show proper reactions to a user's affective state and social behaviour. So instead of forcing ourselves into an interaction that we are not made for, we should rather begin to bridge the social and emotional intelligence gap between humans and machines [323, 324].

3.1.1 Benefits and Applications

The benefits we could gain from affective and socially aware systems are manifold. Some researchers argue that computers will be perceived as more natural, efficacious and trustworthy [324, 325]. That computing will move to background making room for a more human-like and intuitive interaction [238]. And that machines will finally be able to initiate interactions rather than only responding to users's commands and in this way reduce the necessity for a fully attentive and instructive flow of communication [239]. This is backed up by recent findings in cognitive sciences arguing that people's success in life does not only depend on their IQ, but rather their emotional and social intelligence [2, 127, 268, 269]. Certainly, it is not useful to turn every computer into an intelligent social companion and in many situations traditional computer devices do a great job the way they have been designed and always will [321]. However, it is easy to think of applications that would greatly benefit from computer systems which are able to sense and express non-verbal behaviour. And once the technology will be available new ways of using it will be explored for sure.

Interestingly, automatic assessment of human behaviour could be used to gain a better understanding of human-human communication. Manual analysis of human interaction is an enormously time-consuming task and suffers from subjective judgements between raters. Automatic annotation of multimodal data could be very time-saving and at the same time raise the quality of the description [321]. Recordings of group meetings, for instance, could be automatically labelled in terms of dominance, roles, level of interest, and categorized into different actions, like discussions or presentations [324]. In fact, the first pioneering works by Pentland and colleagues dealt with predicting the result of salary negotiations, hiring interviews, and speed-dating conversations [241–243]. Recognition of dominance and personality traits in group conversations has been extensively studied by [8, 16, 33, 155, 230, 232, 247, 251, 262, 355].

From a larger point of view automatic assessment of human behaviour also builds the ground for another recently approaching field called *reality mining* [88]. In Reality Mining data is directly collected from the environment sensed by mobile phones and similar devices and e.g. used for social network analysis. In 2008 Reality mining has been declared to be one of the "10 technologies most likely to change the way we live" by Technology Review Magazine[1].

Another field of application are computer based coaching systems that automatically analyse and judge user performance during certain tasks, such as job interviews or oral presentations [246, 267]. Through audio-visual scene analysis, coaching tools could report and evaluate the social behaviour of individual participants in multiparty meetings [247]. However, a virtual tutor not only has to correctly recognise and interpret the observed behaviour. It also needs the ability to present the lessons in a natural way, e.g. in form of a virtual agent. Again, the expression of non-verbal cues are the key for establishing a trustworthy and convincing presentation [284].

[1] `http://www2.technologyreview.com`

Apart from those and similar visions, there are also rather pragmatic reasons. In the past years computers have evolved into the privileged medium for social interaction between humans [267]. But while information technology increasingly permeates our lives, the interaction medium, e.g. tablet-computers and smart phones, tend to become smaller and start to disappear [239]. This, of course, requires new interfaces that will replace conventional haptic-based interaction. And again, the most obvious solution to this issue is to draw on humans' ability of non-verbal communication and make use of the rich world of affective and socials signals.

3.1.2 From Explicit to Implicit Interaction

In conventional HCI the flow of information begins with the user who wants to accomplish a certain task with aid of a computer. Therefore the user gives explicit commands by clicking buttons, moving sliders, typing text, etc.. For this purpose, mouse and keyboard, and recently also touch sensing surfaces and keyword spotting, are well suited input devices. While we may argue that restricting interaction to explicit input may not serve for a natural and intuitive interaction, grounding interaction on explicit commands has the big advantage that user input is linked to certain meaning. And since the mapping is known to the system it can be directly and unambiguously interpreted. Hence, the user maintains full control of the interaction and is not confronted with inappropriate or unexpected system behaviour (well, at least ideally).

Despite all the mentioned benefits it may bring, involving implicit signals into the interaction also means that there is no longer a definite mapping of user input to system action. While a system will always react to a button being pressed, it may very well overlook a social signal and appropriate responds will fail to appear. Likewise, pressing a button will always trigger a definite respond, but false detections and ambiguous meanings of social cues may lead to unwanted system behaviour. Finally, buttons do not show up from nowhere, whereas unseen, non-prototypical user behaviour may occur and has to be handled in some way. Hence, a socially aware machine not only needs to be equipped with appropriate sensor devices that capture sufficient information about emitted social signals, but it also requires tailored and robust detection tools that are able to extract the encoded cues. And it needs a strategy to assign meaning to them.

3.1.3 From Non-Verbal Cues to Social Signals

Although there is no common definition of the term social signal (see [240] for a detailed discussion), social signals are usually described as temporal patterns of a multiplicity of non-verbal behavioural cues which last for a short time [325] and are expressed as changes in neuromuscular and physiological activity [321]. For instance, we may signal empathy towards a friend by smiling at him (thus the smile becomes a social signal). Our friend may consciously or unconsciously perceive the smile from our face by observing according non-verbal cues, namely raised

lip corners and wrinkles around the eyes. When our friend derives the information it may now change his attitude toward the interaction [242], e.g. he may take it as a sign of understanding.

Sometimes we consciously draw on social signals to alter the interpretation of a situation, e.g. saying something in a sarcastic voice to signal that we actually mean the opposite [267]. But social signals are not always communicative. Social signals, which indirectly convey information about social actions, attitudes, or relationships, are called informative [252]. Mimicry of social behaviour, i.e. when mimicking the verbal and nonverbal expressions of our counterpart, is such an example. In this case, the signal is not a specific action, but the similarity of actions [323]. Such signals are sometimes called *honest signals* since it is difficult to fake them [244]. Turn taking and backchanneling (e.g. nodding), on the other hand, are examples for signals meant to synchronise a conversation and allow a smooth transition between speaker and listener. However, social signals are generally polysemous and the most plausible interpretation may only be derived based on information coming form context [323]. Hence, the emitted non-verbal cues interpreted in front of the current context are the key for understanding social signals and social behaviour.

3.2 Social Signal Processing

A machine that already exists and does an excellent job in detecting and interpreting social cues is the human brain. Social cues are carried through physical quantities and to receive them, the human brain is equipped with a bunch of organs that collect all kind of information from the surrounding: eyes detect light to see what is happening in front of them, ears capture sonar waves to sense sound, the skin senses physical contact, etc.. By constantly picking up these measurements and converting them into electro-chemical impulses our brain can communicate with its environment and people, in particular. In the same way, a machine meant to analyse nonverbal behaviour needs constant measurements of the current world state. That is, it needs to be equipped with adequate sensors, e.g. microphones and cameras, as replacements for the human auditive and visual organs.

Since physical quantities, such as light rays and sound waves, are ever-present and not exclusively reserved for human communication, intelligent algorithms are now needed to separate relevant information from irrelevant one and to give meaning to it. Designing robust and fast-responding systems to detect and interpret social cues is the holy grail in social signal processing. Like the human brain, which is made of specialised regions to fulfill certain functions, it is best practice to split processing into several separate steps. According to Figure 3.1 the core steps of a processing pipeline are:

1. Capture and convert the signal into digital digits.
2. Apply pre-processing to enhance/transform the signal.

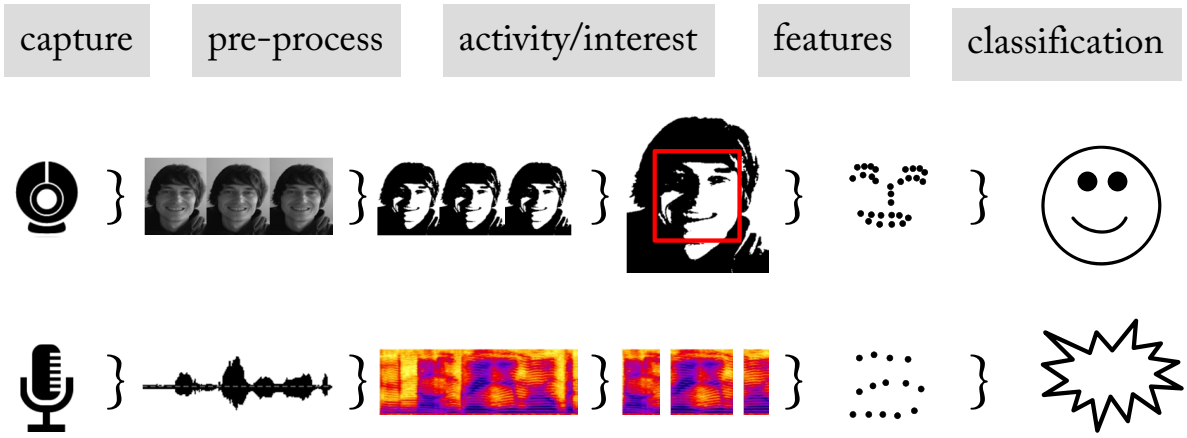


Figure 3.1: Core steps of an audiovisual processing pipeline.

3. Detect activity / regions of interest and extract signal chunks.
4. Convert signal chunks into a compact set of features.
5. Based on extracted features classify chunks into a set of pre-defined categories.

In the following we will discuss these steps by means of two examples: an audio signal carrying affective speech and a video capturing the facial expressions of a user. For the sake of a clear illustration, the procedure is reduced to basic concepts only. In later chapters this simplified view will be extended with additional concepts, such as synchronisation and fusion of multimodal information.

3.2.1 Data Capturing

As a first step, light rays and sound waves need to be captured and converted into a digital signal. A microphone converts sound in air into an electrical signal. With help of an analog-to-digital converter voltage values are converted into a series of digital numbers. Audio signals are characterised by a high sample rate, which should be at least 8000 Hz, i.e. 8000 scans (samples) per second. To cover a frequency rage similar to human hearing, at least 44 kHz should be used.

In comparison 10 – 12 images a human eye can distinguish in one second is a rather small number so that a frame rate of 24 frames per second in a video streams is sufficient to create the sensation of visual continuity [259]. Standard webcams, for instance, operate at up to 30 images per second. The resolution of a video stream is given by the number of pixels per image frame expressed as product of width and height. Typical resolutions are 320×240, 640×480, or 800×600. High-definition video features higher resolutions of 1,280×720 or 1920×1080, sometimes combined with a higher frame rate of 60 frames per second. Typically, frames are

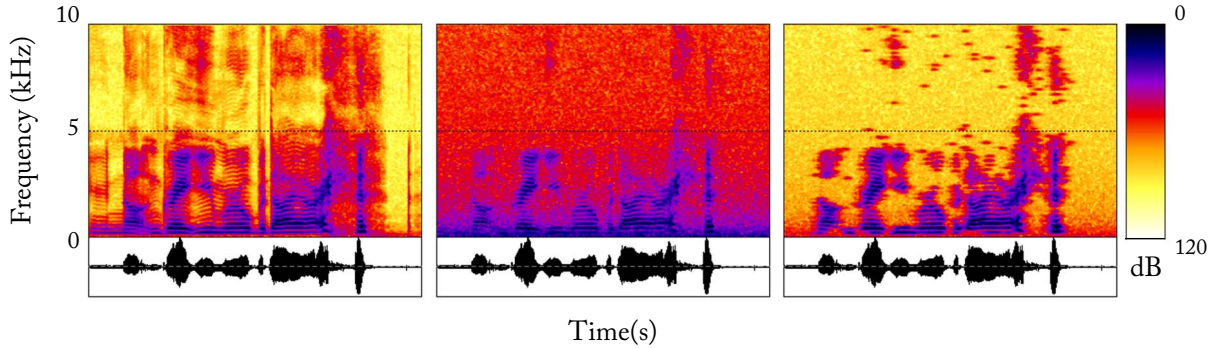


Figure 3.2: Noise-reduction is a typical pre-processing step to enhance the quality of an audio signal. The goal here is to separate a user's voice from background noise and other irrelevant or disturbing sounds. Depending on how aggressively the algorithm removes unwanted frequencies, some real detail will get lost, which can have a negative effect on the further processing. Left: a clean speech signal; Center: original signal overlayed with -20 dB brown noise; Right: the de-noised signal.

transmitted uncompressed in RGB or YUV format. RGB is an additive colour model in which the colour of a pixel is expressed as an triplet of how much red, green, and blue is included. More similar to the human perception is YUV, which defines a colour space in terms of one luma (Y) and two chrominance (UV) components. A grayscale image is an image in which each pixel is mapped to a single value.

Now that we have the raw signals converted into a digital representation and copied to the memory of our machine, the actual fun begins.

3.2.2 Pre-Processing

Pre-processing defines the step during which a raw signal is enhanced and prepared for further processing. This may be less important under controlled conditions, e.g. in a recording studio, but becomes essential in naturalistic settings. A simple, but effective way to remove artifacts, for instance, is to cut off signal values over a certain threshold. However, finding a adequate threshold can be tricky and it might be advisable to adjust it dynamically at run-time, rather than setting it to a fixed value. Also, sometimes the sample rate of a signal is reduced to save further processing time.

Another typical example for early treatments of a signal is noise-reduction [291] and bandpass filtering [256]. In case of noise reduction the goal is to recover the original signal source before further analysis is applied. Bandpass filtering allows it to focus on frequency bands in which useable information is expected. Often, before applying such and similar transformations, a signal needs to be transformed to a another value range or domain, for instance, via normalisation to a common scale or by transforming the signal into the frequency domain. The Fast Fourier transformation (FFT) allows to decompose a time series into components of different frequencies

Figure 3.3: The contrast of the original image (left) is increased (middle); edges are easier to detect (right).

[86]. Since many properties of a signal are best recognised in frequency domain frequency domain analysis and Fourier transforms are cornerstones of signal analysis.

Figure 3.2 depicts the application of a noise reduction algorithm to a speech signal overlayed with brown noise. Since the goal is to remove unwanted frequencies, the operation is applied in frequency domain. In video analysis, it is common practice to convert coloured images into grayscale. Increasing the contrast of an image is useful to carve out edges. Figure 3.3 gives an example. An important feature of a filter is the ability to work on a short moving window, which allow for an incremental processing of a signal. Otherwise, it might not be applicable in real-time systems.

3.2.3 Activity / Area of Interest Detection

Knowing *when* and *where* in a signal relevant information occurs is extremely important. Failure or success of further processing steps highly depends on this stage, as now it is to decide how to split the signal into meaningful chunks. A useful unit to judge the paralinguistic properties of a speaker, for instance, would be the beginning and ending of a spoken utterance. To this end, a voice activity detector decides for each frame whether it contains silence or speech. The decision can based on energy, zero crossing, pitch, and other statistical measures [204]. Neighboured voiced frames are likely to belong to the same utterance and should be grouped before further analysis is applied. Apart from that, is saves processing time and improves the robustness of a system since now frames without activity can be ignored a priori (see Figure 3.4).

Reducing the spatial information within a sample, e.g. an image, is another way of cutting content to shape. Facial expression analysis, for instance, starts by detecting areas that include a face. Other parts of the image are not of interest and can be skipped. Since face detection has to deal with interpersonal variations as well as changes in scale, orientation, lighting conditions, occlusions, etc. , it has aroused the interest of researchers for quite some while and several approaches are found in literature ranging from *knowledge-based* methods, which try to detect limbs by applying pre-defined rules derived from human knowledge, over *feature invariant*

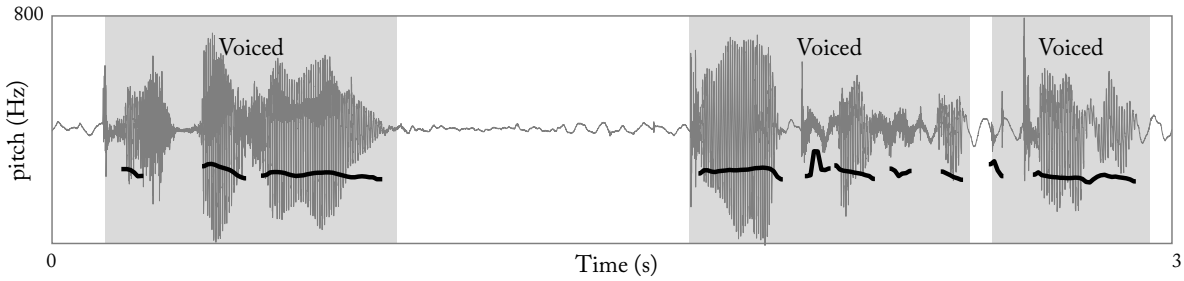


Figure 3.4: Determining which parts of a signal should be grouped is essential to build meaningful chunks. In the example, pitch values are extracted from a waveform to separate voiced and unvoiced frames in the signal. Voiced frames can then be combined for further analysis, while frames without activity will be ignored.

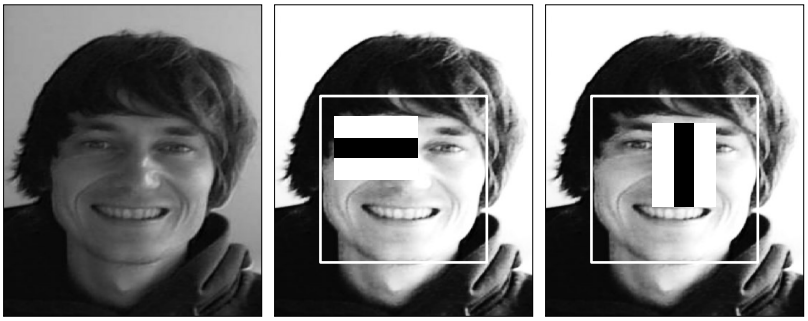


Figure 3.5: The face detection algorithm by Viola and Jones [326] uses Haar-filters (middle and right picture) relative to the detection window (white box). The sum of the pixels which lie within the white rectangle are subtracted from the sum of pixels in the grey rectangle. The algorithm has become popular due to its robustness in terms of orientation and lighting conditions.

approaches, which search for structure features that are robust to pose and lighting variations, *template matching* methods, which compare image regions with pre-stored body templates, and *appearance-based* methods, which learn models from a database showing the object under a large number of viewpoints and illumination conditions [353].

A popular and robust face detector meant to work in real-time under realistic conditions has been designed by Viola and Jones [326]. It is based on so-called *Haar-like features*, which are calculated over a sliding rectangle and capture intensity differences within the observed regions. Since certain parts of a face, e.g. the eyes, tend to be darker than other areas, e.g. the cheeks, a classifier will learn patterns that are characteristic for a face (see Figure 3.5). To speed learning classifiers are arranged in a cascade and successive classifiers are trained only on samples which pass through the preceding classifiers – a variant of the so called *AdaBoost* algorithm, which combines several weak classifiers to build a strong classifier. As long as each classifier is slightly better than random it will improve the final model [210]. Since its publication in 2001, many extensions have been made to the original algorithm, such as introducing features with more variations [196] or replacing the standard *AdaBoost* scheme with *RealBoost* [350]. Experiments show that such systems and humans make similar face detection errors, which suggests that

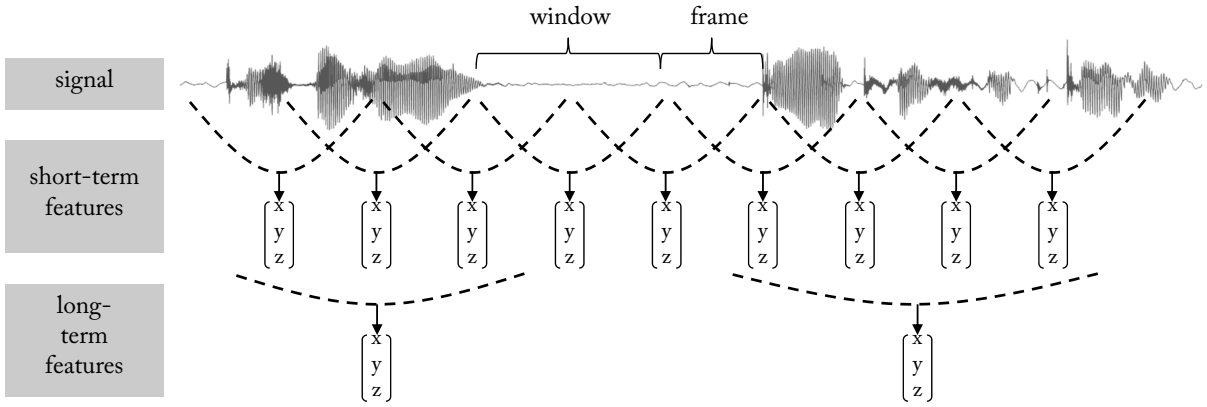


Figure 3.6: Short-term features are extracted over a shifted window of fixed length. The obtained time-series of feature vectors may serve as input for following feature extraction steps. If at later stages features are computed for long windows of several seconds we denote them as long-term features.

detection relies on similar features [143]. A survey by Zhang and Zhang provides a detailed description of the Viola-Jones algorithm and summarises many of the recent advances in face detection [360].

3.2.4 Feature Extraction

Generally spoken, a *feature* transforms a signal chunk into a characteristic property. Choosing discriminating and independent features is key to any pattern recognition algorithm being successful in classification. For instance, calculating the energy of an utterance will reduce a series of thousand measurements to a single value. But whereas none of the original sample values is meaningful by itself, the energy allows for a direct interpretation, e.g. a low energy could be an indication of a whispering voice. By this means features help to carve out information relevant to the problem to be investigated, while at the same time the amount of data is considerably reduced. However, usually it is not a single, but a bunch of features that are retrieved, each describing another characteristics of the input signal. It is common practice to represent features by numerical values or strings and group them into *feature vectors*.

Features that reduce an input sequence to a single value are *statistical* features or *functionals*. Typical functionals are mean, standard deviation, minimum, maximum, etc.. But sometimes, the original sequence is also transformed into a new, although shorter sequence. Such features are called *short-term* features. To extract them a window is moved over the input sequence and a feature vector is extracted at each step. Often, successive windows overlap to a certain degree. In this case, the *frame size*, also called *frame step*, defines how much the window will be moved at each step. Often, the extraction of short-term features is only an intermediate stage which leads to another feature extraction stage. The relationship is visualised in Figure 3.6.

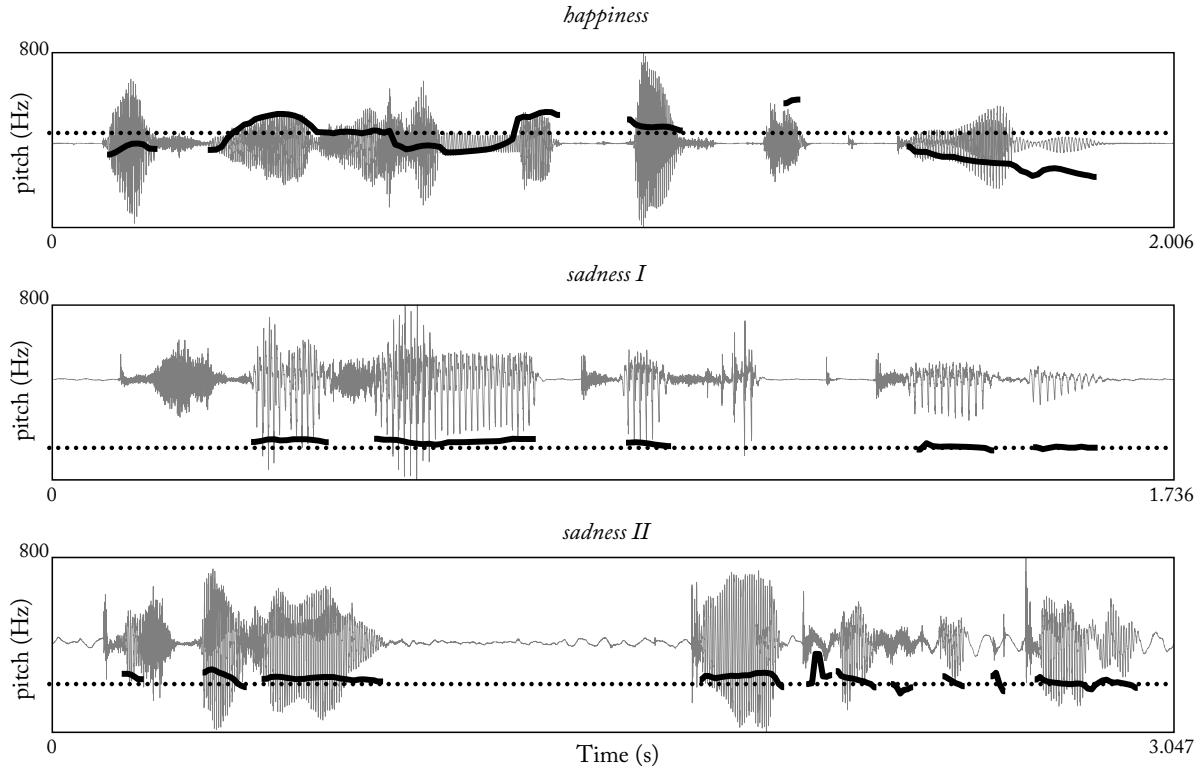


Figure 3.7: Although the wording is the same for the three sentences, they differ in their pitch contour. If articulated with a happy voice (top graph), there are more variations in the pitch contour and in average pitch values are higher compared to sentences pronounced with a sad voice. But even if emotional expression match, there are variations in the pitch contour due a different speaking timing (compare the two bottom graphs). By taking the average of the contours (dotted line) local changes are discarded and their relationship becomes more obvious.

A "good" feature encodes information that is relevant to the classification task at hand. In other words, there is a sort of correlation between distribution of feature values and the target classes, which allows reasoning the class from the feature. Figure 3.7 shows the pitch contour of three utterances articulated with either a happy or a sad voice. Although the semantic content of the sentences is the same their pitch contour is different[2]. This also applies for examples belonging to the same class, which makes it difficult to compare the sentences. By taking the average of the contours local changes are discarded and it becomes more obvious which utterances draw from the same class.

In automatic emotion recognition features most features are derived from *loudness*, *pitch*, *voice quality*, *spectrum* and *Mel-frequency cepstral coefficients* (MFFCs). The loudness of a sound is based on its amplitude and correlates with high arousal. A low variation is a hint for bored speech. Pitch is a perceptual phenomenon which determines how "low" or "high" a sound is perceived. Happy speech, for instance, is characterised by a higher contour and greater variance.

[2]"Das will sie am Mittwoch abgeben" ("She will hand it in on Wednesday"), taken from "Berlin Database of Emotional Speech" [37]

	Intensity (*dB*)				Pitch (*Hz*)				Spectrum ($Pa^2/Hz \times 10^3$)			
	mean	std	min	max	mean	std	min	max	0-1k	1k-2k	2k-3k	3k-4k
happiness	76.0	14.4	32.8	86.5	244.9	54.6	144.7	365.9	25.3	4.60	0.46	0.32
sadness I	76.0	12.2	37.8	84.1	104.5	9.4	84.9	118.3	24.6	0.49	0.18	0.25
sadness II	79.8	11.2	44.5	88.1	154.8	20.1	122.3	246.6	109.6	0.76	0.56	0.22

Table 3.1: Functionals for the audio samples in Figure 3.7. Several functionals are applied to the intensity and pitch contours, namely arithmetic mean (*mean*), standard deviation (*std*), minimum (*min*) and maximum (*max*) value. In addition, the mean power of four frequency bands has been extracted from the spectrum.

However, it is important to consider that pitch alters differs greatly between subjects, especially between males and females. For male voices it typically ranges between 80 and 180 Hz, and for female voices from 150 to 300 Hz. Pitch determination has to face several problems concerned with the determination of voiced/unvoiced parts, rapid changes of pitch with time and falsely detected multiples of the "true" fundamental frequency, known as harmonics. Methods for pitch extraction in time domain use zero crossing points or the autocorrelation, which measures the correlation of a waveform with itself. Also a number of frequency domain approaches exist which are filter based or use cepstrum or multi-resolution analysis [122]. Voice quality is an umbrella term for a bunch of features describing the colouring of a voice, such as *jitter* and *shimmer*. Jitter and shimmer are measures of the cycle-to-cycle variations of pitch and amplitude. They are commonly measured for long sustained vowels. High shimmer and jitter values are usually perceived by humans as breathy, rough or hoarse voices [112]. However, they have also shown value in stress and emotion classification [194]. The spectrum is a representation of a speech signal in the frequency domain. It can be derived by applying the short-time Fourier transform (STFT), i.e. each frame is Fourier transformed to calculate the magnitude of the frequency spectrum. The spectrum highs, identified by spots with high energy, are called *formants* and represent the vocal tract resonances. Since different positions of the formants result in different sounds they play an important role for the generation of different phonemes in human articulation. Yet, some studies report correlations with emotional speech production. For example, Biersack and Kempe [28] associate higher first formants, and Waaramaa *et al.* [329] higher third formants with positive emotions. France *et al.* [117] found features derived from formants to carry cues to depression and suicidal risk. Although originally developed to filter out non-linguistic effects, MFCCs have proven to yield good results in emotional speech recognition tasks [23, 181, 189, 224]. MFCCs are derived from the cepstrum, which is calculated from the spectrum [70]. Similar to the way the spectrum separates a signal in repetitive time patterns, the cepstrum maps harmonic series to the same coefficient. To better approximate the human auditory system the frequency bands of the spectrum are positioned logarithmically according to the Mel scale [308].

Table 3.1 presents statistical features derived from pitch, intensity and spectral features for the

three speech signals in Figure 3.7. As mentioned, the quality of a functional depends on its ability to measure class-dependent variance. The arithmetic mean (*mean*) of intensity, for instance, does not differ between sadness and happiness (row *sadness* and row *happiness*) and hence does not contain sufficient discriminative information. The other values, i.e. standard deviation (*std*), minimum (*min*) and maximum (*max*), are more promising candidates as they differ between the two examples. Taking into account the second sadness example (row *sadness II*), we would pick standard deviation of intensity, the four pitch features, and energy in the second (*1k-2k*) and fourth spectrum band (*3k-4k*), as the most significant features since values are smaller for the two sadness samples compared to the happiness sample. However, to assess their actual value more samples are needed. In the end, it is the combination of several measurements which allows a reliable separation of the classes.

In order to categorise face images according to their expression, features are needed that describe deformations of certain facial points as related to their neutral position. Especially for detecting subtle changes temporal dynamics may be important, too, which can be received by analysing series of successive images. Here, the work by Paul Ekman and in particular the *Facial Action Unit System* (FACS), which has been developed under his supervision [92], is to mention. Their motivation for creating a standardised set of facial parameters was to replace human observers in facial behaviour researchers as they may be biased by context, e.g. accompanied speech or due to a different cultural background. The basis of FACS are Action Units (AUs), which encode visible actions performed by single or groups of muscles. AU1, for instance, is the action of raising the inner brow, which is caused by the *frontalis* and *pars medialis* muscles. There are also AUs dedicated to eye movement (e.g. AU61, which stands for "eyes turn left") as well as head movement (e.g. AU51, which stands for "head turn left"). FACS also provides intensity scores annotated by appending letters A–E: A=trace, B=slight, C=marked or pronounced, D=severe or extreme, E=maximum. Using FACS facial expressions can be precisely explained by annotating involved AUs. A shortcoming of the FACS system, however, is the poor time coding. Extensions like FACS+ proposed by Essa and Pentland try to overcome this limitation [105].

3.2.5 Pattern Recognition

Last but not least, a mapping from signal chunks, also called *samples*, to a set of target classes is required. Instead of discrete categories sometimes one or more continuous values are chosen. The choice depends on the particular problem to be solved. In emotion recognition tasks, for instance, it depends on the emotion model that is used to describe the affective content. Targeting discrete categories goes well with an emotion model based on discrete entities, while a continuous classification fits a dimensional emotion model. In any case, decisions are derived on basis of the extracted features, i.e. either a single vector composed of long-term features, or sequence of short-term feature vectors. According to the type of features a *static* or *dynamic*

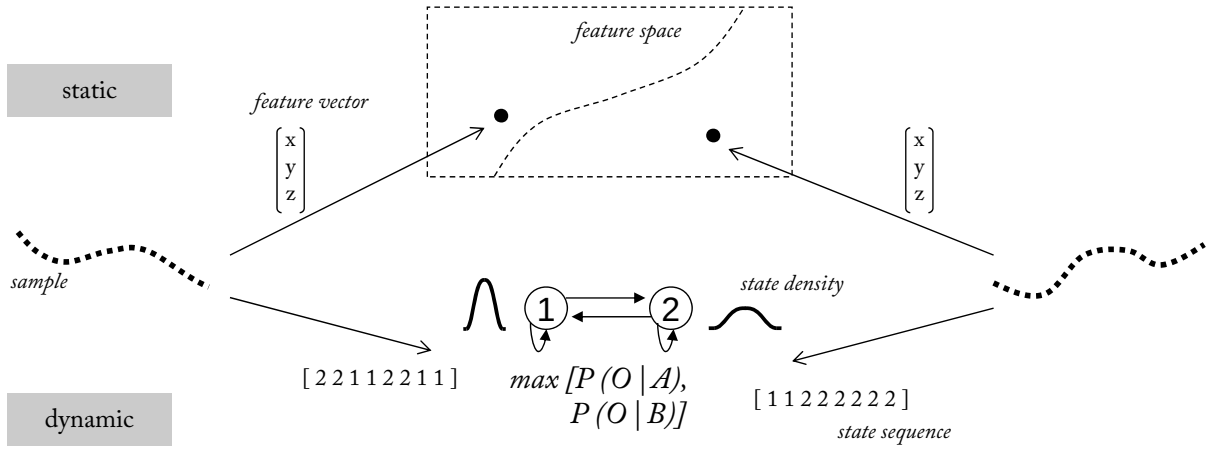


Figure 3.8: In static modelling (top) a sample is reduced to a single feature vector and classified according to its location in the feature space. In dynamic modelling (bottom) classification is derived directly from the contour of the sample, e.g. by describing it as a sequence of state changes and deciding in favour of the model that returns the most plausible description.

classifier is needed. In case of static modelling the similarity of two samples is expressed by their distance in the feature space. In dynamic modelling decisions are derived from the raw sequence of low-term features, which are not necessarily of same length. Here, similarity of two samples is calculated based on their shape contours (see Figure 3.8).

A fair number of *pattern recognition / machine learning* approaches have been proposed for automated *classification*. They all start from a learning phase in which they are presented with training samples in order to tune a model that allows them to generalise to an arbitrary input sequence. If the training algorithms does not require class information, i.e. which sample belongs to which class, we call it an *unsupervised learner*. In this case similar samples are automatically grouped into clusters. The most popular method following this approach is the *k-means clustering*, where training samples are portioned into k clusters by minimising the distance between the samples of a cluster and its centroid. However, unsupervised learning does not guarantee that found clusters will match desired classes. Hence, most classification algorithms adopt *supervised* learning.

Figure 3.9 shows an example of a simple facial expression recogniser. A database is created with images showing faces in a *happy*, *neutral*, and *sad* mood. Images are grouped according to their class labels and represented by two features: one value expressing the opening of the mouth and a second value measuring the distance of the mouth corner to the nose tip. Since position and shape of the mouth are altered during the display of facial expressions, we expect variations in the features that are distinctive for the target classes. Identifying possible variations in the training samples and embedding them in a model that generalises to the whole feature space is the crucial task of a classifier. Figure 3.9 shows the position of each sample in the feature space that is spanned by the two features. Here, we can see that samples representing the same class

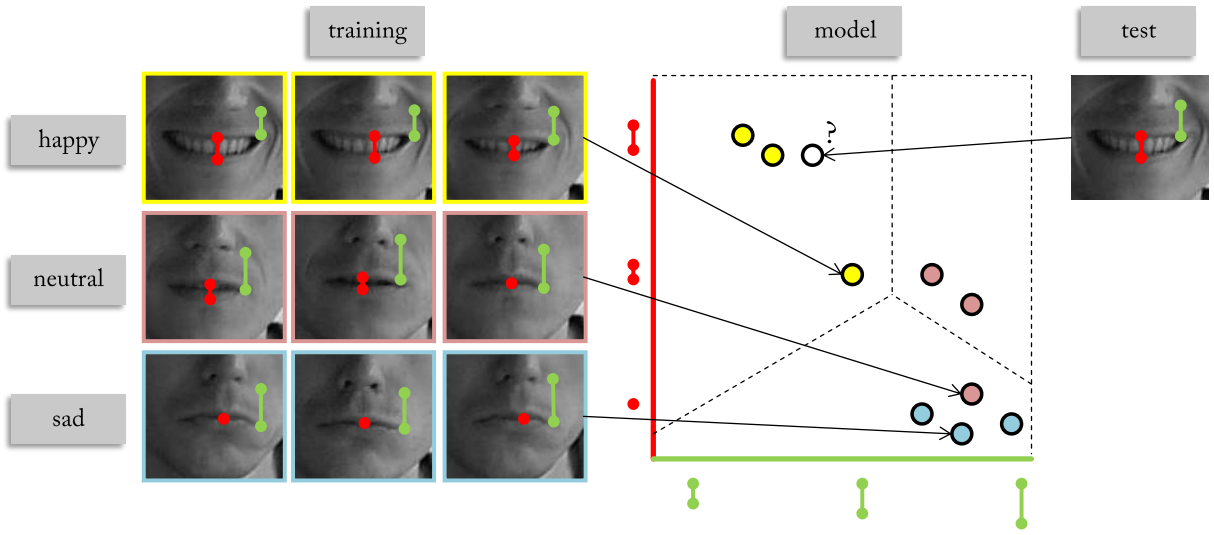


Figure 3.9: In a facial expression recognition task samples are selected from the area around the mouth in four classes: *happy*, *neutral* and *sad* (left). Each sample is represented by a feature tuple – opening of the mouth and distance of the mouth corner to the nose tip. Based on the training samples a linear model is learned to separate the feature space according to the three classes (middle). Unknown samples are classified according to their position relative to the decision boundaries in the feature space (right).

tend to group in certain areas of the feature space. In the concrete case, *happy* samples – mouth opened and lip corners raised – end up top left, whereas sad samples – mouth closed and lip corners pulled down – are found in the right down. Distance between mouth corner and nose tip turns out to be of similar for *neutral* and *sad* samples, so that they are only distinguishable by the opening of the mouth. Based on the distribution of the samples the feature space is now split in such way that samples of the same class preferably belong to the same area. The boundaries between the areas are called decision boundaries.

It is important to note that the aim is not to find a perfect separation of all samples, but to built a good hypothesis in respect to unseen samples. According to the decision boundaries in Figure 3.9, for example, one of the *neutral* samples actually lies within the boundaries of the *sad* class and hence would be treated as a *sad* sample. However, adjusting decision boundaries to perfectly fit the training samples can be dangerous since it is likely that outliers will be given too much meaning, probably leading to a higher number of false detection when applied to the test data. Hence, the crucial task of a classifier is to find a good generalization from the training samples to unseen data. The following listing introduces some popular classification algorithms.

Decision Tree A tree with each branch composed of an exclusive decision according to a single feature and leaves representing the final classification. In order to keep the tree as small as possible most relevant features are usually located at the top of the tree. A *Random forest classifier* uses a number of decision trees, in order to improve the classification rate [216].

K-Nearest Neighbour (KNN) A very simple learning technique where samples are classified based on k closest examples (*instance-based learning*). The function is only approximated locally and all computation is deferred until classification (*lazy learning*) [57].

Linear Discriminant Analysis (LDA) Belongs to the family of linear classifier and tries to find a linear combination of features which separates the classes best (*Fisher analysis*). For the purpose of classification the feature space is separated by linear planes [85].

Support Vector Machine (SVM) Belongs to a family of generalised linear classifiers which construct a maximal separating hyperplane to separate the classes (*maximum margin classifier*). Maximum-margin hyperplanes are found by looking for the largest 'tube' that can be drawn around a decision boundary not containing samples. Samples along the hyperplanes are called *support vectors* [249].

Gaussian Mixture Model (GMM) Distribution of the input values are modelled with a number of *probability density functions* (pdf), usually Gaussians. A pdf is similar to a histogram, but provides greater flexibility and precision in modelling the underlying statistics of sample data. Once a model has been generated for each class, an unknown sample is classified according to the class which model gives the highest conditional probability, i.e. explains the sample best [260].

Hidden Markov Models (HMM) A simple network composed of a finite number of states, transitions between those states, and output symbols. At each time step a single output symbol is emitted by the model, and only this sequence of observations is actually visible to the observer, while the underlying state sequence, which has produced the outcome, is hidden. Given a set of competitive models a certain sequence of output symbols is classified according to the model which most probably has generated the sequence [257].

Artificial Neural Network (ANN) Interconnected group of artificial neurons arrayed in *input layer*, one or more *hidden layers*, and *output layer*. Input values are propagated through the network from layer-to-layer, whereby the strong linking between adjacent neurons allows modelling of complex relationships between inputs and outputs. Among many variants *feedforward neural networks*, which move information only in forward direction, are most commonly used [29].

3.2.6 Deep Learning

Recently, a new branch of machine learning has evolved called *deep learning*. The roots of deep learning go back the 1980s when researches began to design brain-inspired systems, namely ANNs, to learn own rules from scratch. In that regard deep learning differs fundamentally from

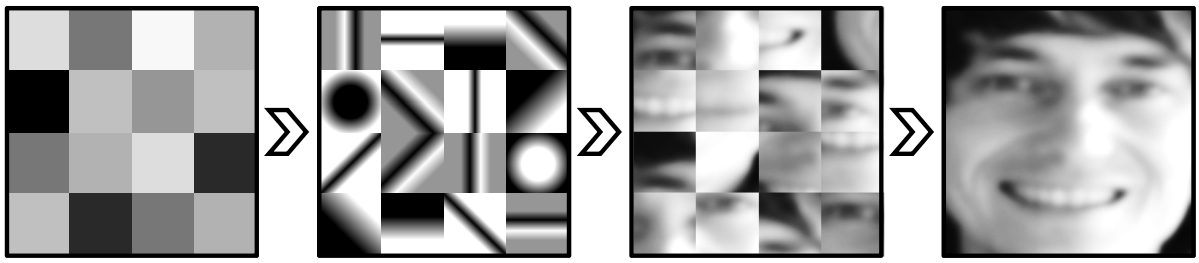


Figure 3.10: Deep learning uses a layer-based structure to extrapolate from simple shapes to complex objects. At a first stage pixels of different hues are identified. These are combined to form edges and simple shapes which provide the bases to recognise more complex objects. Finally the model learns which objects can be used to define a human face.

traditional methods, which require the extraction of task-dependent features. In deep learning, however, feature engineering becomes obsolete by promoting the principle of raw-feature-based end-to-end learning [76]. As a matter of fact, intermediate representations are automatically derived using a layered structure in which higher level concepts are learned lower level ones (see Figure 3.10).

For one reason, the comeback of deep learning is due to the increases in computing power, which allows simulating a net of 1 million neurons interacting through 1 billion connections, as in case of *Google Brain*, a deep learning research project at Google. For another reason, it is because of the explosion of digital data [161]. One of the first successes was reported by Mohamed *et al.* [217, 218] who applied *deep belief networks* for the task of phone recognition. Their system outperformed traditional methods such as HMMs on the TIMIT benchmark database. Le *et al.* [188] showed impressively that with help of deep learning it is possible to train a face detector from only unlabelled images. After all, their dataset consisted of 100 million images taken from the internet. To train their network the researchers employed a cluster of 1,000 machines with 16,000 cores for three days. The final model was robust not only to translation but also scaling and out-of-place rotation. And it was able to learn other high-level concepts such as cats or human bodies. In 2012, Krizhevsky *et al.* [183] won the ImageNet competition with an error rate of just 15.3% compared to 26.2% by the the second-best entry. Task of the contest was to classify 1.2 million images into 1,000 categories. Their neural network consisted of 60 million parameters and 650,000 neurons organised in five convolutional layers.

Motivated by these success stories deep learning has been applied in other areas, too, such as speaker recognition [305], drug target prediction [315], three-dimensional image analysis [303] and malware detection [67]. In the field of affective computing deep learning algorithm has been applied to the recognition of emotion from speech [1, 139, 193], video [162, 163, 263] and physiological data [201], although the underlying databases are considerable smaller compared to those used in speech recognition or image retrieval tasks. To exploit the full potential of deep learning for social signal processing access to large, real-world data sets will be needed. Another

major barrier to the application of deep learning is that it currently requires considerable skill and experience to choose sensible values such as the learning rate or the number of layers and units per layer [76]. Guessing good model parameters is especially crucial since training is extremely time consuming. Propagating an unknown input sequence through a trained network, on the other hand, is very fast, which makes it applicable for real-time processing. Hence, we can certainly expect that deep learning will play an increasingly important role for social and affective signal processing in the future.

3.3 Two Paralinguistic Studies

Paralinguistics may reveal two types of information: user *states* that change over time, e.g. emotional colouring, and permanent user *traits*, such as age and gender [288]. Both types are relevant for nonverbal behaviour analysis. The first because it encodes information about the current affective state and the latter as it provides important context how to correctly interpret detected cues. To provide a test-bed for comparing approaches towards the analysis of paralinguistics in naturalistic speech and language, since 2010 a special issue called the *Paralinguistic Challenge* is held at INTERSPEECH[3]. Every year the organisers of the challenge provide large datasets with distinct definitions of test, development, and training partitions, incorporating speaker independence and different miking, as needed in most real-life settings. Researchers from all over the world are encouraged to participate and provide their results on specific tasks. In the following, two contributions are presented. Both of them follow the core steps of a machine learning pipeline, but implement some specifics in order to improve the performance of the final classifier.

3.3.1 A Frame Pruning Approach

The first study [334] was submitted to the *Personality Sub-Challenge* as part of INTERSPEECH 2012 Speaker Trait Challenge [295]. The provided corpus includes 640 clips from 322 individuals in five categories: OPENNESS, CONSCIENTIOUSNESS, EXTRAVERSION, AGREEABLENESS, NEUROTICISM. Each sample is connected to a set of 5 labels that denote which categories $X = \{O, C, E, A, N\}$ apply. If a category does not apply it is marked with a leading N. Samples are provided in three sets: a training set (n=256) to train the algorithm, a development (n=183) for evaluation, and an unlabelled test set (n=201) to participate in the challenge. Task is to correctly predict the personality of the speaker.

[3]INTERSPEECH is the world's largest and most comprehensive conference on issues surrounding the science and technology of spoken language processing, both in humans and in machines

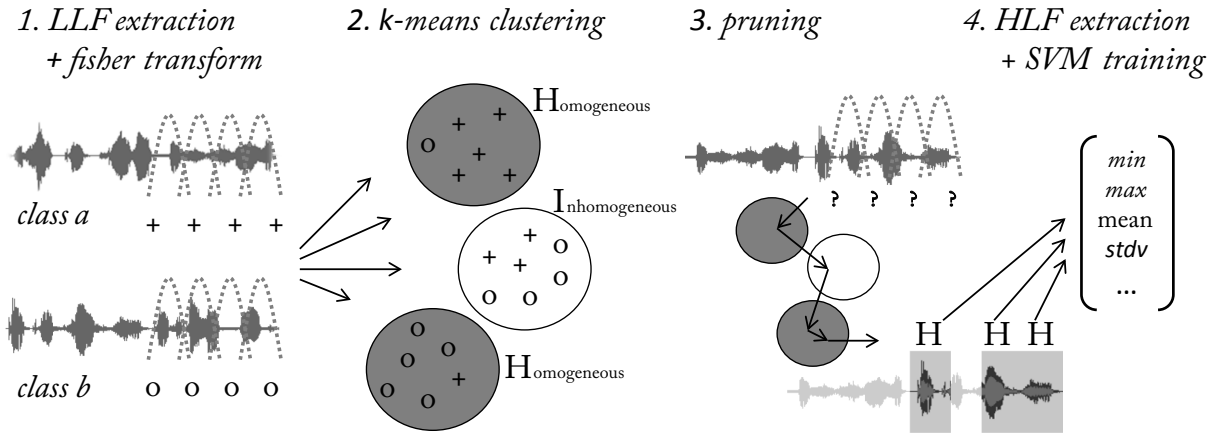


Figure 3.11: First, low-level feature are extracted for each frame and Fisher projection is applied (1). Next, transformed frames are clustered using K-Means. If the majority of frames inside a cluster belong to the same class we regard the cluster as *homogeneous*, otherwise we mark the cluster as *inhomogeneous* (2). Low-level feature vectors are then pruned by cutting out frames belonging to *inhomogeneous* clusters (3). Finally, high-level features are extracted from the pruned samples and used to train a SVM classifier (4).

Approach

As described earlier, in speech analysis voice activity detection is applied to split the raw audio stream into discrete chunks. Then, features are extracted for the whole chunk. This procedure presumes meaningful information to be embodied within the whole segment. This assumption may be misleading if distinctive cues within a sample are surrounded by non-meaningful information or noise. In this case it would surely be beneficial to keep only parts of the sample that are most relevant for the recognition task. Hence, a novel cluster-based approach is proposed which aims at identifying frames likely to carry distinctive information. It is inspired by a technique applied in speaker verification tasks. To find the most likely speaker in speaker identification tasks, it is common practice to average frame scores over the whole test utterance. However, Besacier *et al.* [27] decided to apply a hard threshold to sum only frame scores that are most likely to identify a speaker, which led to a significant improvement of identification rate.

Algorithm

The first step of the algorithm concerns the segmentation of the input samples into smaller parts. Low-term features are calculated at a frame step of 10 ms using TUM's open-source openSMILE feature extractor [107, 109]. The set is based on energy, spectral and voicing features (see Table 4 in [295]). Together with delta regression coefficients, for each frame 129 low-level features are extracted and linked with the class label of the according sample.

To speed up the following clustering step Fisher projection [85] is applied to the frames in the training set in order to find a transformation into a feature space with less dimensions (here 2, as it is a 2-class problem). Since Fisher projection uses *Linear Discriminant Analysis* (LDA) to minimise the variance within classes and maximise the variance between classes, a better class separation is obtained for the target space, which increases the probability to find *homogeneous* regions at a smaller number of clusters. Then, K-Means clustering [13] is applied in order to group frames into well separated clusters[4] – yet independent of any class affiliation. Frames in the same cluster are alike, those belonging to different clusters show differences.

For each cluster the class distribution is calculated among its members. This is achieved by counting the frequency of frames belonging to the same class, divided by the total number of frames inside the cluster. Here, the assumption is that the higher the relative amount of frames belonging to the same class, the more meaningful information is carried by the members of the cluster. Those clusters are called *homogeneous*. Clusters that show a more or less equal class distribution, on the other hand, are not likely to host observations significant for any of the classes. Such clusters are regarded as *inhomogeneous*. To decide whether a frame is to be regarded *homogeneous* or *inhomogeneous*, the following definition is applied: *If at least $T\%$ of the observations of a cluster are linked with class c, the cluster is regarded as homogeneous with respect to c, otherwise it is treated non-homogeneous.*

Figure 3.12 illustrates the result of that rule applied to the NEUROTICISM task ($K = 500$ and $T = 70\%$, i.e. a cluster is regarded as *homogeneous* if at least 70% of the frames inside the cluster belong to the same class). After dividing the features space in *homogeneous* and *inhomogeneous* regions, for every frame it is decided, whether to keep it for the classification process or not. This manifests the actual pruning step. Therefore, for each frame the nearest cluster is determined. Frames belong to a *homogeneous* cluster are kept, all other frames are discarded. Since long sequences of *homogeneous* frames are sometimes interrupted by only a single or few *inhomogeneous* frames each decision is replaced by a majority vote among its neighbours.

Finally, the pruned sequences are prepared for final classification by extracting a set of long-term features. Again the openSMILE tool [107, 109] is used here to extract a final set of 6'125 features (see Table 5 in [295]). However, instead of the raw speech files pruned sequences are used as input. As classification scheme a Support Vector Machines (SVM) with Sequential Minimal Optimisation (SMO) and a polynomial kernel is employed as provided by the WEKA data mining toolkit [138].

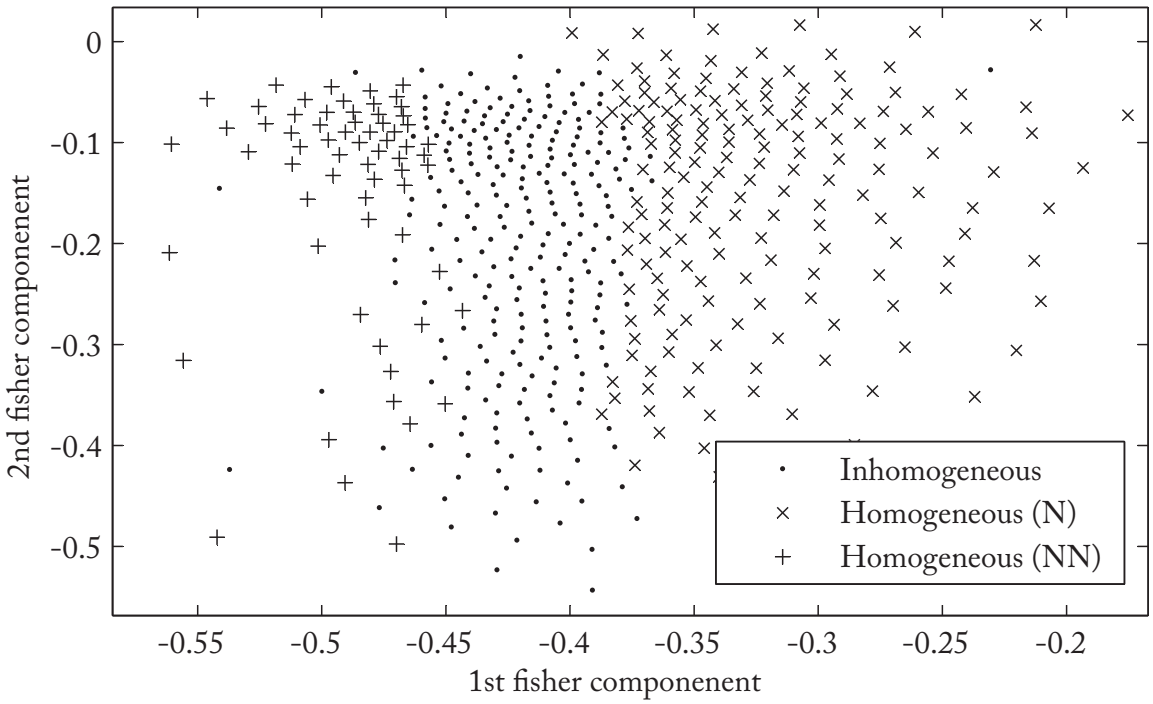


Figure 3.12: Distribution of *homogeneous* and *inhomogeneous* clusters for NEUROTICISM (*K=500* and *T=70%*). *Homogeneous* clusters are found at the right and left edges, *inhomogeneous* clusters accumulate in the center, where classes overlap.

Results

If the proposed algorithm is successfully carving out cues that are distinctive for the recognition task, a classifier trained on the pruned sequences should achieve a better accuracy compared to a classifier trained on the original samples. Since the strength of the pruning depends on the chosen parameters results several combinations are tested to find an optimal configuration. Table 3.2 compares results for original and pruned files, reporting a an increase for the latter by 3-4%, which proves the usefulness of the approach.

3.3.2 Using Phonetic Patterns for Detecting Social Cues

The second study [335] was submitted to the *Social Signals Sub-Challenge* as part of INTERSPEECH 2013 Computational Paralinguistics Challenge [296]. It sets the task to localise and classify laughter and fillers in the "SSPNet Vocalization Corpus" (SVC) composed of 2'763 audio clips collected in 60 phone calls involving 120 subjects. Laughter and fillers like "uhm"

[4]K-Means clustering is a popular method because of its simplicity and stable performance – n observations are partitioned into k clusters in which each observation belongs to the cluster with the nearest centroid.

	UA recall in %	
	ORIGINAL	**PRUNED**
O	56.3	62.1
C	70.1	71.3
E	80.4	82.0
A	60.4	61.7
N	65.1	67.6
mean	**66.5**	**68.9**

Table 3.2: Recognition results on development set. Pruning configuration: $T = 70$ and $K = 1000$.

and "ah" are social cues expressed in human speech. Detection and interpretation of such non-linguistic events can reveal important information about the speakers' intensions and emotional state [325]. Laughter, for instance is often a sign of amusement or joy, although it may also show scorn or embarrassment [266]. Fillers, on the other hand, are indicators to hold the floor or signal that what was said will be revised by the speaker [54].

Approach

Although paralinguistic analysis generally is not interested in decoding the semantic context of a speech message, phonetic information can still provide useful hints to the classification process. This study investigates ways to enrich a standard paralinguistic feature set with phonetic transcriptions. Of particular interest to this work is a study by Dutoit and Urban [316] who performed a phonetic transcription of laughter. They found that individuals use rather different subsets of laughter phonemes. On average, about thirty phonemes per laughter occurred in the corpus they investigated. Since several sounds could not be found in the International Phonetic Alphabet, Dutoit and Urban used additional labels for the manual transcription of laughter, such as groan, snore, and grunt. Although Dutoit and Urban focus on laughter synthesis, their work provides useful insights for laughter detection as well. However, the transcription of laughter has been conducted manually in their work. Regarding fillers, it is a common approach to augment speech recognition with according models to detect so called Out Of Vocabulary (OOV) [87]. Today's speech recognisers usually come with pre-trained models for such sounds.

Algorithm

To create a phonetic transcription for the raw audio files, a phoneme detector is applied as a preliminary step. To this end version 3 of the *CMU Sphinx* toolkit for speech recognition [5]

[5] `http://cmusphinx.sourceforge.net/`

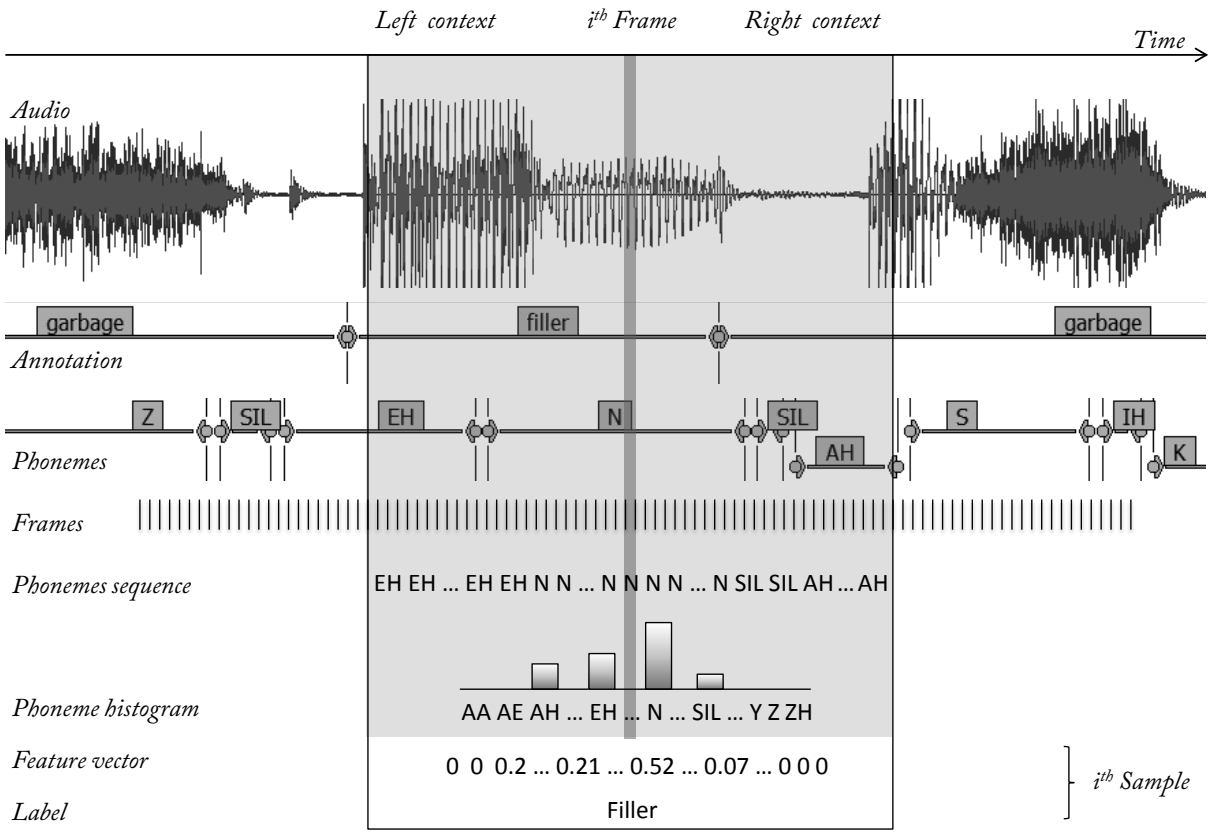


Figure 3.13: Extraction of phonetic features: first, for each frame the sequence of phonemes is determined by mapping the phonetic transcription on frames of 10 ms length. The context size n defines how many frames to the left and right are taken into account. Then, a histogram is built by counting the occurrence of each phoneme in the sequence. Finally, the relative phoneme frequencies are stored as features labelled by the class of the center frame.

is employed. It is used as a sort of a black box with a standard phoneme model that was not particularly tuned on the observed data. After having retrieved a phonetic transcription for each audio file in the database, transcriptions are mapped on a set of low-term features, which describe the distribution of phonemes within the surrounding of a frame.

As illustrated in Figure 3.13 a histogram is calculated which stores the frequency of each phoneme within a certain range. This range is defined by the frame itself plus n frames to the left and n frames to the right. Hence, n controls the context that is used to extract the features. Obviously, the length of the feature vector equals the number of phonemes (and fillers) in the dictionary: here 48 features. Finally, feature values are normalised by dividing them by the context length $(2 \times n + 1)$, so that feature vectors sum up to 1. This feature set is denoted as *pho-1*.

Since a histogram stores only the frequencies at which phonemes occur, while no information about the temporal ordering is kept, important temporal cues might get lost, e.g. on the rhythmic of laughter or that a filler is represented by a certain order of phonemes. To capture those cues a second type of feature is implemented: instead of calculating a single histogram over the whole

garbage		laughter		filler	
SIL	7406	+C+	672	N	894
IH	6521	SIL	157	SIL	608
N	5457	N	117	AE	523
AH	5316	T	106	EH	398
T	5049	IH	94	D	339

Table 3.3: Counts for the 5 most frequent phonemes per target class at segment level in the training set. Note that due to lack of space +COUGH+ has been shortened to +C+.

context, two histograms are extracted – one for the left context (including the current frame) and one for the right context. In this way differences in the phonetic distributions on the left and right context of a frame can be measured. Obviously, this doubles the number of features in the set. This feature set is called *pho-2*.

For classification again a linear kernel Support Vector Machines (SVM) with Sequential Minimal Optimisation (SMO) as provided by the WEKA data mining toolkit [138] is employed. To train the classifier features are first extracted on the complete training set (1'735'770 frames) and downsampled to achieve a balanced class distribution (79'572 frames). The trained model is then tested on the development set (547'789 frames). Accuracy is measured in terms of the Unweighted Average Area Under the Curve (UAAUC) for the laughter and filler classes at frame level.

Phoneme Distribution

If it is possible to distinguish laughter and filler from garbage frames using the proposed feature differences in the frequency at which certain phonemes occur with respect to the classes are to be expected. Table 3.3 lists counts for the 5 most frequent phonemes per target class. It can be seen that the most frequent phoneme occurring within garbage is the phoneme for silence (SIL). This reflects the fact that large portions of the audio files actually contain silence, e.g. when the user is listing to the operator. However, SIL turns out to be a frequent phoneme within laughter and filler segments, as well. Probably because both vocalisations are likely to be surrounded by silence as they interrupt the normal speech flow.

Regarding laughter findings are actually in line with the study by [316], although they conducted a manual transcription using a phoneme set adapted to laughter. The SIL phoneme, which was the most dominant phoneme after the cough sound, was the phoneme most frequently observed by them in laugher exhalation phases. Also, the non-central vowel I and the nasal phoneme N were among their most frequent phonemes. Dutoit and Urbain found the large number of the plosive T a bit surprising, an observation which could, however, also be confirmed by the study. However,

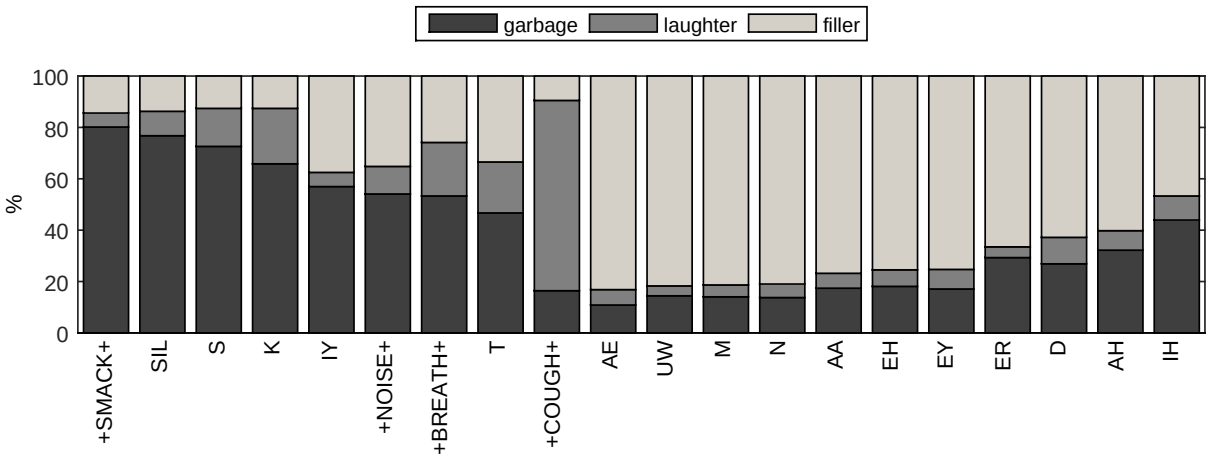


Figure 3.14: Relative frame frequency per class for the 20 most frequent phonemes (90.1% of total mass) on the downsampled training set (no context).

a similar large number of the breathy phoneme HH as reported in [316] was not found because the Sphinx dictionary included a filler +BREATH+ onto which breathy sounds were mapped.

So far annotations have been observed at segment level. This might be misleading, because even if a phoneme occurs regularly within two classes the duration at which it occurs may still differ. Figure 3.14 compares occurrences of the 20 most frequent phonemes (90.1% of total mass) among classes at frame level. To have a fair comparison numbers are obtained for the downsampled training set where the number of frames per class has been balanced. In fact, a clear preference of some of the phones for certain classes is observed. For instance, phoneme +SMACK+ has a 80% probability to occur during garbage frames. Likewise the phonemes AE or M are popular filler phones, probably reflecting the sequence {AE M}, which is a popular filler pattern. The most prominent phoneme correlating with laughter is still +COUGH+. It seems that the filler model for coughing actually shares very similar phonetic properties with laughter events in the corpus.

Results

The tendency of certain phonemes to correlate with certain classes gives grounds to believe that a classifier can be trained to predict target classes above chance level. To test the usefulness of the approach, a baseline (*base*) feature set is extracted and combined with the two phonetic features sets (*base* + *pho-1* and *base* + *pho-2*). Therefore 6'373 paralinguistic features are extracted using TUM's open-source openSMILE feature extractor [107, 109]. The set includes energy, spectral, cepstral (MFCC) and voicing related low-level descriptors (LLDs) as well as a few LLDs including logarithmic harmonic-to-noise ratio (HNR), spectral harmonicity, and psychoacoustic spectral sharpness.

Results are summarised in Table 3.4. It shows an improvement of about 5% when phonetic

	UAAUC in %		
	base	*base + pho-1*	*base + pho-2*
laughter	86.2	88.2	93.3
filler	89.0	83.1	92.0
mean	87.6	92.5	92.7

Table 3.4: Recognition results for different feature sets on the development set: *base* = baseline features, *base + pho-1* = baseline features and phonetic features extracted on single histogram, *base + pho-2* = baseline features and phonetic features extracted on two independent histograms.

features and baseline features are combined to a single set. However, the choice of the context length has an significant influence on the quality of the phonetic feature set and should be large enough to capture sufficient information about neighbouring phonemes. In the reported experiments results become stable for $n > 50$, which corresponds to a window length of more than one second (which is actually roughly the average length of a laughter segment).

Chapter 4

Challenges and Lessons

In the last chapters we have learned about the important role social and affective signals play in human communication and what are the benefits of a human-computer interaction which no longer ignores the social context. Also, we have discussed techniques that allow a machine to automatically extract non-verbal information. Nevertheless, we must notice that despite a great deal of work that has been carried out in this area, the undertaken effort has not translated well into applications. In this section we search for reasons why this is the case and what can be done to tackle pending issues. The identified shortcomings have been the driving force for the development of a novel framework named SOCIAL SIGNAL INTERPRETATION (SSI). SSI provides a multimodal framework for recording, analysing and fusing social signals in real-time and will be introduced in detail in the following chapters. To draw a bow we will conclude each section with an outlook to what extent the described issues have dropped into the development of SSI.

4.1 Collecting Large Multimodal Databases

Studying social interaction means studying humans, and humans are different from each other. They differ in their personal values, they show varying emotional expressivity, have been raised in front of a certain cultural background, and develop different personality traits, to name only a few properties that make people individuals. Furthermore, the environment in which social interaction takes place is everything but stable. It can be in a quiet room or a noisy street, at daylight or in a disco with flashing lamps all over the place, at a familiar talk between two friends or at a cocktail party with plenty of people talking across each other, etc. Last but not least, there are ethical reasons which demand that a person must not be recorded without consent. Hence, the recording of humans that are involved in affective and social interaction can be an enormously time consuming task. Nevertheless, it is one of the most crucial steps towards building automatic

detection systems. After all, it is the collection of representative training samples which defines the knowledge base any derived model can draw on. Or to phrase Roddy Cowie, one of the pioneers who would spend big efforts on collecting such databases: “a neural network can only achieve what its training data allows it to” [60].

4.1.1 The Risk of Overfitting

Let assume your boss instructs you to train a system to automatically separate pictures of smiling and neutral faces. In order to collect a set of training images you go out on the streets and you ask passersby to allow you taking pictures of their faces. The first day you tell them to look neutral and you use those pictures as examples for a *neutral* class. The next day you ask everybody to smile and this way you collect samples for a *smile* class. Next, you employ an artificial neural network (ANN) to which you feed a fraction of the images together with the information whether they show a neutral or smiling face. After the learning phase is completed you use the remaining photos to validate whether the network is able to correctly distinguish the images. You are happy to see that the output of the system is correct in all cases! When you show it to your boss he gets suspicious and instructs you to collect another set of images. After another day out on the streets you come back and feed the new photos into the system – and this time the output is completely random. At first, this looks like a mystery, but after reviewing the images in the training set you suddenly notice that that all the images with neutral faces have been taken on a cloudy day while all the images with a smiling face have been taken on a sunny day. Obviously, the system has learned to separate picture taken on a cloudy days from pictures taken on a sunny day.

Although it is only a fictitious example, researchers have actually experienced similar situations [84]. The lesson to learn is that even if a learning algorithm has been trained to distinguish positive and negative examples at an acceptable success rate, this is no guarantee that the learning was actually successful. In supervised learning a classifier automatically tunes itself from a set of training samples for which input and output is known. Afterwards it is able to make predictions for unseen samples, i.e. for any input an output is generated. But the fact that most of the powerful classifiers work as black boxes is a blessing and a curse at the same time. It is a blessing because such a classifier can be applied to any learning problem that we are able to describe in numbers, e.g. the question whether an image – represented as a matrix of colour values – contains a neutral face (0) or a smiling face (1). And it does not even demand of us to have a clear idea how a mapping between input and output can be achieved. We just throw some data at it and see what happens. But it is exactly the *see what happens* that should keep us in suspense whether the learning was successful or not. An ANN, for instance, updates internal weights to fit a given target function. But once training is completed, it is (almost) impossible to tell what exactly has been learned by the net, e.g. whether it has learned to distinguish neutral and smiling faces or something else. The only benchmark a system has is the desired target

function. How the mapping will be achieved is left to the learning algorithm. On the one hand, this is surely an advantage compared to rule based approaches, which will always require some sort of expert knowledge. On the other hand, there is a risk of learning something unwanted. The risk increases with the number of model parameters and the dimensionality of the feature space. And it becomes even worse if only a small number of samples is available to the learner. In this case, the model will likely suffer from what is called *overfitting*. Meaning it will show a good performance on the training set, but generalise poorly on unseen data. And such a model will almost certainly fail when put into practice. In fact, to have any practical value a model must be able to generalise from the observations it has been presented during training to a virtually infinite number of unseen samples. But is there are a way to make sure that this will be the case? Not really, we can only take some precautions.

A common way to avoid overfitting is by splitting the corpus into sets for training, development, and testing. Then, optimise the model on training and development set, and finally evaluate it using the test set. Unfortunately, even this strategy cannot guarantee reasonable performance on unseen data. On the one hand, the model may learn peculiarities of the corpus, e.g. it may be tuned to certain properties that are common to all or at least most samples in the corpus, but are not necessarily met in the real-world. On the other hand, the model may fail in situations where it is used to make a prediction on samples that were simply not represented in the databases, e.g. a model trained on a group of male subjects can show a good performance for other males, but fail when applied to females. In the ideal case the model will identify them as outliers, but even this information has to be derived from the training units. The most promising way to avoid the mentioned problems is by using comprehensive training corpora stocked with sufficient variations to cover as many contingencies as possible. And as long as training data can be easily generated, e.g. via simulation, this is a reasonably fair task. When it comes to the study of human behaviour, however, it is no cakewalk.

4.1.2 From Exploration Towards Applications

Most studies in the past (and not only in the past) have analysed social and affect behaviour on data that had been recorded by professional actors (sometimes also non-actors) in controlled lab settings [20, 37, 101]. Obviously there are some good reasons to rely on acted data. For instance, it is way easier to instruct someone to perform a particular set of actions than trying to induce the same behaviour within an expensive experiment. Not only will it be costlier to collect an adequate number of samples; the observed behaviour will almost certainly be expressed on a broader scale, i.e. include more variations with a strong trend towards subtle and less prototypical manifestations. In addition, signals obtained within a less controllable environment are rather prone to artifacts due to noise, occlusions, movements, etc. Although both variations in behavioural expressions and the variations in signal quality, come nearer to the truth, the effort

to collect a sufficient number of samples as well as the complexity of the recognition task, will increase to a considerable degree. Hence, it is fair to ask whether the additional costs that are caused by moving from acted to natural interaction are really worth the pain. Yet, there is some strong evidence that this is indeed the case.

One of the first studies meant to systematically investigate the effect of genuineness of the training data on recognition performance goes back to Batliner *et al.* [17]. Objective of their work was the implementation of an automatic dialogue system to be used in call-centres. Such systems are cheap to maintain but bear the risk to (repeatedly) misunderstand and/or misinterpret spoken instructions. Obviously, this must be a frustrating situation for the calling customer. In the worst case he will be so upset that he hangs up. Since everything is better than losing a customer, it would be preferable to have the system determine when a customer is about to terminate the dialog in order to pass it over to a human operator. Expression of anger or irritation in the voice of the customer could be a reliable sign to initiate a fallback. And from what had been reported in literature by that time, Batliner and colleagues had good reasons to believe that this should be a fairly simple task. It turned out it was not.

Early studies on judging emotional states from a user's voice date back to the 1930s. In general, they follow a similar procedure: speakers (often professional or semi-professional actors) are asked to give vocal portrays of different emotional states by producing a set of utterances. The recorded samples are played back to judgers, who have to guess the encoded emotion. The observed set of emotional states are for the main part fundamental emotions, (e.g. joy, sadness, anger, etc.). In order to exclude semantic information either nonsense or emotionally neutral sentences are used, or masking techniques are applied to eliminate the intelligibility of speech. A comprehensive review by Scherer [272] reports an average accuracy of about 60%. A considerable higher result than we would expect by chance, which is reported to vary between 10% and 25%. Knowing that the paralinguistic part of a vocal message discloses reliable information on the affective state of the speaker, researchers also tried to identify relevant acoustic cues. With few exceptions, e.g. the recording of a radio reporters' emotional commentary on the explosion of the Zeppelin "Hindenburg" in 1937 [341], acted material forms the primary source of such studies. A good overview is again given by Scherer in [274]. Anger, for instance, is described by an increase in pitch, intensity, and high-frequency energy. Apart from that, pitch is reported to be downward directed and the rate of articulation is supposed to rise. Sadness, on the other hand, would be characterised by a decrease of pitch, intensity and high-frequency energy as well as a dropping pitch contour and a slow down in the rate of articulation.

To come back to the problem tackled by Batliner and his colleagues [17], there seems to be a straightforward way to derive the decision whether to continue a conversation or hand it over to a human operator: extract according prosodic features and apply the rules reported in literature. This approach appears to be specially promising since the target system only has to make a binary decision, i.e. it has to decide if the current caller's voice contains signs of anger or not. Apart

from that, callers are likely to share a similar vocabulary and sentence construction. Nevertheless, what seemed to be easy at a first glance, turned out to be quite complicated. Unfortunately an observation that has to be made all too frequent when the objective of a study is not only to evaluate an algorithm on some pre-recorded data, but actually putting it into practice, i.e. to set up a system that would handle input on the fly in a realistic situation outside the lab. And one reason can be found in the often too optimistic assumption that observations derived from datasets that are not eligible to depict reality at a sufficient level will without further ado generalise to the real world. For instance, actors learn to express emotions in a caricatural way, whereas affective signs as we use them in everyday communication are rather subtle [80]. Likewise, most studies concentrate on a set of of basic emotions displayed in a pure manner. But so called full-blown emotions can rarely be observed in natural interaction, which is rather dominated by blended emotions [80]. Hence, high recognition rates reported for experiments with skilled speakers that act *as if* they were in a specific state of arousal may not necessarily be transferred into the world outside the lab.

To gain a better insight into the nature of the problem, Batliner and his colleagues [17] decided to run tests on three separate datasets. Each set would contain the same kind of affective content, yet different methodologies were used for their elicitation. The first database compromised samples from speakers that were asked to imagine situations in which the dialogue system fails. For the second database, a subset of the sentences contained in the first database was taken and subjects were asked to read them with an appropriate voice. Finally, the third database was collected in a Wizard-of-Oz (WOZ) scenario. A WOZ scenario is an experimental simulation, during which participants are given the impression that they are interacting with a fully automated system. However, it is actually the experimenter who surreptitiously controls the program and moderates the interaction according to his interest [167]. In the concrete case, users were confronted with a dialogue manager that would produce output according to a fixed schema. In this way users were given the impression of interacting with a malfunctioning automatic speech processing system. Although subjects still pretend to need some information, the samples collected in such a scenario are supposed to come closer to reality than purely acted or read data. However, as noted by the authors, such an arrangement will still have the problem that users tend to show more tolerance towards the system than they probably would do in real life.

Next, Batliner *et al.* [17] extracted a set of acoustic and word-based features at utterance level (one sample per utterance) to which they applied standard machine learning techniques. Task of the classifier was to make a binary decision whether a test sample, i.e. an utterance that was not part of the training set, should be considered as emotional or not. Without going into the technical details, we can say that the differences in classification rates for the three datasets were incredibly huge. On the acted data a recognition accuracy of almost 100% could be achieved – problem solved, one may conclude! For the read data, however, the hit rate was only about 80%. And for the samples collected in the WOZ scenario, which are most likely to resemble

reality, results dropped by another 10%, yielding no more than approximately 70%. While this is still considerable better than chance, the shift from acted to naturalistic data is remarkable. Of course the authors were interested in figuring out what are possible reasons for this obvious divergence. By looking at which features contributed most to the classification process they found that speakers in the acted corpus expressed emotions mostly via duration by lengthening important keywords, while in the read scenario emotions were triggered mostly via energy. This suggests that different kind of cues are used to convey emotion in acted and read speech. In the WOZ data, however, prosodic marking could not be assigned to a specific feature class at all. The very convincing explanations given by the authors is that in a WOZ scenario it is left to the speakers how to express emotion, i.e. whether to use prosody alone or another strategy, possibly a mixture. This is a remarkable observation as it implies that increasing the realism of an experimental setting at the same time decreases our ability to make assumptions about the way subjects will express their feelings [18]. In any case, the lesson shows that the progress from exploratory research towards real application relies heavily on the development of appropriate databases [80].

4.1.3 Building Appropriate Databases

By the beginning of the new millennium, experiences like the one just described, made researchers think of novel strategies towards creating a new type of database for modelling human behaviour. In particular it meant turning away from what has been regarded as a classic database design, with Ekman and Friesen's database of emotional faces leading the way. Traditionally, assembling databases had not been considered a very outstanding task and was mainly driven by the search of prototypical samples of human behaviour in preferably pure form. Only with growing experience and the realisation that acted data alone cannot reflect non-verbal behaviour as it occurs in daily life a rethinking of this policy began to take hold. A group of psychologists at the Queen's University in Belfast led by Roddy Cowie and Ellen Douglas-Cowie were among the first paving the way for a new generation of databases. Not only did they collect the first databases of that kind, they also formulated guidelines how to build them [60, 61, 80–82]. From their work we can conclude that four main issues should be considered in developing a database - the scope, naturalness, context and the kinds of descriptors [80].

The *scope* of a database is determined by the number of subjects, their cultural background, age, gender, etc. as well as the kind of behaviour, affective expressions, and actions that are represented. Existing corpora, for instance, are dominated by subjects from Western Europe, mostly middle-aged with an academic background. Or as Henrich *et al.* [144] summarise: from Western, Educated, Industrialised, Rich, and Democratic (WEIRD) societies. What is often happily ignored is the fact that these WEIRD people represent a thin slice of humanity and, to top it all off, a particularly unusual one. Hence, it must not to be taken for granted that

findings from this narrow slice do automatically generalise to the whole species. Same counts for the coverage of social behaviour, which is often understood as a narrow set of prototypical expressions. In case of emotions, for instance, a set of "basic" states (anger, joy, sadness, etc.). Again, this material does not cover the richness of blended and moderate expressions perceived in daily life [61]. Yet, as claimed by Cowie and colleagues [60], it is the responsibility of the community to accumulate data that grasps the phenomenon in its full complexity. As a matter of fact, it requires both, databases moving towards a full coverage of a large number of core states, and databases focusing on a single or few expressions at varying intensity levels.

It is the origin of the material that defines the *naturalness* of a database. Douglas-Cowie *et al.* [80] distinguish between material that is acted, deliberately induced, or culled from existing sources. To induce spontaneous reactions one may employ a WOZ scenario similar to the one described earlier, but other methods have been reported, such as Velten mood induction [169], emotive music [277], affective pictures [186], or games [159]. Broadcast material is regarded as a popular source to study truly natural behaviour, e.g. TV talk shows. Although it is generally preferable to target natural behaviour, we must not forget that this also means to lose control in two ways: on the one hand, it remains entirely open what kind of reactions will be observed in the reviewed material, consequently additional effort has to be spent to establish reasonable ground truth. On the other hand, dealing with the data proves to be more difficult due to background noise, changing camera angles, varying distances from microphones, etc. Apart from that, for ethical reasons publication of such material becomes particularly sensitive. A possible middle way could be the use of samples coming from a real-life situation to guide the artificial production by actors [80]. A thought which has been elaborated by Busso and Narayanan [38]. In their view the key problem is not the use of actors per se, but the ad-hoc elicitation method used in the recording. For instance, instead of asking actors to read sentences, two or more actors could be involved in a loose conversation. The latter is supposed to produce more genuine material and resemble spontaneous speech production to a higher degree. Established acting techniques applied by skilled actors and rehearsing the material in advance may help to avoid exaggerated and caricatural behaviour. To conclude we can say that the preferred strategy depends on the kind of behaviour to be investigated. A lab environment might be an appropriate place to have a discussion between people if the objective is to study turn taking. On the other hand it is very unlike to face a real threat to life under lab conditions [80]. If one has a specific application in mind, setting up a WOZ scenario might be a smart move to learn which reactions are expected to occur.

The need to provide adequate context information has been overlooked for long. Typically, there are two general types of context [61]: temporal context and intermodal context. Temporal context is important at two levels. On a short-term basis it is often the time course of a social cue that conveys the meaning rather than its absolute values. To incorporate this information into a model we can either rely on adequate statistical features (variance, slope, etc.) or apply contin-

uous classification schemes (e.g. Hidden Markov Models or Short-Long-Term Recurrent Neural Networks). On a longer term, temporal context can be considered by interpreting behavioral cues on the basis of prior events rather than in isolation. This, of course, implies that recordings are long enough to make room for a natural evolvement of social signals. A common flaw in emotion corpora, for instance, comes from the fact that affective content is represented in form of short and isolated episodes. While this appears to be handy at a first glance, it does not capture the ebbs and flows of emotions over time [80]. A person showing signs of happiness (usually) will not fall into a deep depression within the next few seconds. Taking this into account allows to build models that are less prone to false detections. And it may help to solve ambiguity, e.g. a smile by an angry person is rather meant to be sarcastic than a sign of happiness. Apart from providing sufficiently long samples, intermodal context is another important source of information. As discussed in detail in Chapter 2, humans are used to express their intensions through a bunch of channels, namely, voice, face, gestures, etc.. However, the interplay is highly complex and there is no straightforward mapping between channels. While sometimes one channel may give additional evidence for an observation in another channel, it is just as likely that multimodal cues will clash [81]. In fact, we can think of situations were it is precisely this discrepancy that provides the decisive clue towards a correct interpretation of the observed behaviour. For instance, the difference between a smile denoting pleasure or amusement, which is not to be confused with a similar expression of anxiety known as a grimace. Or a conscious wink to signal that we actually mean the oppositive of what we seemingly express. A system that overlooks such small but important correlations will obviously not be able to derive a correct assessment of the situation. Hence, the success of an automated analysis also depends on its ability to bring together seemingly contradicting signs. The establishment of sophisticated techniques that are able to put all available information into a larger and consistent picture remains as one of the main challenges to the automatic analysis of social interaction. Although a more detailed discussion is to follow we can conclude here that it is certainly not sufficient to build on databases specialising to a single channel. Instead, databases are required that provide interaction by means of multiple sources as well as descriptions of when and how certain modalities come into play [60].

According to Douglas-Cowie *et al.* [81] the final feature of a database concerns the kind of *description* that is given. In case of acted material this is usually straightforward since a solid description can be directly derived from the experimental setup. When it comes to more naturalistic, possibly continuous material, however, annotation defines a challenge on its own. In fact, is is not uncommon that creating adequate annotations takes way longer than the recording itself. For one thing, inducing behaviour rather than relying on specific commands means losing control if and when a desired situation is going to happen. Hence, raters have to be hired who will manually define the on- and offsets of social episodes and describe them according to a predefined annotation scheme. For another thing, agreeing on an annotation scheme in the first place becomes more difficult since subjects are no longer restricted in the use of interactive strategies,

leading to more mixed and intermodal expressions. In the field of affective display and social signals this is further complicated by the diversity of conceptual ideas. The variety of definitions and methods, which have been proposed to establish a good understanding of human behaviour, have not yet converged to a commonly accepted theory. This is especially problematic, since people with different background are supposed to come to work together interdisciplinarily [240]. For instance, faithful descriptions may be too detailed to have practical value for statistical modelling. Or combining data from several sources is hampered if different annotation schemes have been applied. Yet, the development of commonly accepted measurement methods still remains a core challenge [242]. At a very basic level two fundamental trends can be distinguished, namely categorical descriptors and continuous variables [81]. The preferred choice often accrues from the phenomenon to be described, e.g. it seems natural to attach a symbolic gesture (icon) with a specific label, whereas the fluidity of a gesture is best characterised by a continuous value. To describe higher-level mental states and behaviour, however, the choice becomes less obvious. As discussed in Section 2.4.2 emotions are a typical example where descriptions of both kind can be applied. As long as full-blown emotions are at the focus of attention, they can well be described by a limited set of categories. But, when it comes to blended emotions applying a dimensional representation might be a more adequate choice.

4.1.4 Available Databases

One of the first databases meeting the new design criteria proposed by Douglas-Cowie and colleagues [80] has been published by the authors themselves. The Belfast Naturalistic Database [79] contains 239 audio-visual recordings from a total of 100 subjects. While it is still a relative small number of samples, the content of the database differs from other databases at that time. Most clips are extracted from broadcast material and contain mixed emotion rather than archetypal examples. Relying on a large number of subjects guarantees a variety of alternate expressions for the same kind of emotional state as they are likely to appear in everyday communication. In terms of annotation, the database incorporates two types of description – dimensional and categorical – so that developers can choose the one that suits their objectives. Since then, several multimodal corpora with emotional content, but a significantly larger amount of data, have been released to the community. E.g. the SmartKom corpus [280] containing 172 recordings of subjects asked to test a system "prototype" for a market study that in fact was controlled by two human operators, the VAM corpus [131] consisting of 12 hours of recordings of the German TV talk-show "Vera am Mittag" (Vera at noon), the SAL (Sensitive Artificial Listener) corpus [83] consisting of 11 hours of induced emotion using an interactive artificial personality, the IEMOCAP (Interactive Emotional Dyadic Motion Capture) database [40] gathering 12 hours recordings of dyadic sessions where actors perform improvisations or scripted scenarios, and the SEMAINE corpus [202] composed of 100 sessions each about 5 minutes recorded during

Name	Subjects	Duration	Continuous	Neutrality	Media
Conflict	135	12 h	no	natural	audio, video
MAHNOB	40	13 h	yes	mixed	audio, video
GEMEP-FERA	7	25 min	no	roleplay	video
CK+	123	10 min	no	scripted	video
Idiap Wolf	36	7 h	yes	mixed	audio, video
USTC-NVIE	> 100	natural	video, infrared		
SEMAINE	20	12 h	yes	natural	audio, video
Belfast Naturalistic	125	no	natural	audio, video	
FeeTalk	20	4 h	yes	natural	audio, video
HPEG	10	no	scripted	video	
Humaine	no	mixed	audio, video		
Canal 9	350	45 h	yes	natural	audio, video
HCRC Map Task	64	16 h	yes	roleplay	audio
IDIAP Head Pose	16	yes	natural	video	
Green Persuasive	16	5 h	yes	roleplay	audio, video
ICSI Meeting	61	72 h	yes	mixed	audio
AMI Meeting	189	90 h	yes	roleplay	audio, video
MMI-Facical Experession	95	4 h	no	mixed	audio, video

Table 4.1: Databases hosted on *SSPNet* (retrieved April 2015). Half of them are multimodal, although exclusively audiovisual.

emotionally coloured conversations.

While the mentioned corpora specifically address emotions, there is fair number of multimodal databases with focus on other social phenomena. Many of them have been collected within meeting and game scenarios, offering insights into turn taking strategies, role allocation (e.g. dominance), and personality traits. To name a few, there is the AMI Meeting Corpus made of 100 hours of meeting recordings [45], the Canal9 corpus providing roughly 42 hours political debate [322], the Idiap Wolf data set containing around 81 hours of conversational data among groups of 8–12 people playing a role playing game [147], and the MAHNOB Mimicry Database assembled of 13 hours interaction of in total 40 participants discussing on a political topic or attending a role-playing game [195, 310].

The collection of adequate databases is a complex and lengthy undertaking, which can be achieved only through collaboration within the community. To this end, the *SSPNet Portal*[1] hosts a repository of multimodal corpora, which are listed in Table 4.1. It is notable that most of the databases contain 10 *h* of interaction and about half of them include continuous content. Also, many were collected in a natural or at least semi-natural (i.e. role-play) environment containing at a dozen subjects and more. However, it also strikes that, although half of corpora are multimodal, the only media are audio and/or video. Other modalities such as 3d body and gaze tracking,

[1] `http://sspnet.eu`

or physiological signals are not present, but are needed to provide a broader picture of human interaction.

4.1.5 Call for New Databases

The increasing interest in building multimodal corpora is evidenced in a series of International Workshops on Multimodal Corpora[2]. The aim of this periodic event, which has become an integral part of large international conferences (e.g. LREC), ranges from collection efforts, coding, validation and analysis methods, to tools and applications of multimodal corpora. Between 2008 and 2010 the number of accepted paper in that track has doubled. A survey paper by Ěerekoviæ summarises the most relevant submissions [89]. Ěerekoviæ concludes that despite the effort that has been spent on building comprehensive multimodal database containing meaningful examples of social behaviours, there is still a lack of data. Since the richness of observable social expressions make it impossible to obtain a single corpus for all social phenomena, data should be collected in as many different scenarios as possible. This concerns the location where the interaction takes place as well as the roles of involved subjects, conversation topics, etc. Here, games and competitions seem to be the most promising candidates to observe naturalistic behaviour within a still reasonably controlled environment. Apart from that the selection and placement of the sensors that are used to record the interaction are of importance. Although Ěerekoviæ notes an increase of recording equipment other than high-quality video cameras, 70% of the reviewed databases are still audio-visual only [89]. Yet, this is not only due to a lack of alternatives. Microsoft's Kinect and comparable depth sensors are good examples for a growing market of novel recording equipment that is easy to use and cheap. The reason that most current corpora are still restricted to audio-visual content comes from the difficulty to synchronise different sensor devices.

And although several hours of interaction sound like a considerable amount of data, it is still small compared to databases available for speech recognition, of which some have a vocabulary size of more than 200 billion tokens [50]. This is due, among other things, to the fact that automatic speech recognition benefits from a well established annotation scheme, namely phonetic transcriptions, so that speech databases can be merged with relatively little effort. Whereas in the field of social signal processing, still no coding scheme exists that would fully cover all aspects in human interaction behaviour. And even where a certain level of standardisation has been achieved one may still find oneself faced with the choice of several conflicting models, for instance whether to describe emotional content by discrete labels or dimensional coordinates (see Section 2.4.2). As a matter of fact, researchers either adapt existing schemes to their needs or build them from scratch. Unfortunately, a large part of the proposed multimodal schemes

[2] http://http://www.multimodal-corpora.org/

lack proper validation and have proven to be unreliable [47]. A few exceptions to this rule exist, though. The most prevalent standard is the well known FACS (Facial Action Unit System) developed by Ekman and Friesen to describe the richness and complexity of facial expressions [92]. An example for a truly multimodal annotation scheme would be MUMIN by Allwood and colleagues [3]. MUMIN deals with gestures and facial displays in interpersonal communication, with particular regard to the role played by multimodal expressions for feedback, turn management and sequencing. An annotation scheme for marking subjective content in meetings, specifically the opinions and sentiments that participants express as part of their discussion, has been introduced under the acronym AMIDA (Augmented Multiparty Interaction with Distance Access) by Theresa Wilson [343]. Likewise, Maptask [44] and DAMSL (Dialog Act Markup in Several Layers) [55] define annotation schemes for describing communicative acts in dialogs. To overcome the limitation of most existing coding schemes to deal with only one or two modalities, Blache *et al.* [30] propose to combine existing schemes and extend them to obtain a coding scheme that would be as complete as possible. One such approach is PAULA XML (Potsdam Exchange Format for Linguistic Annotations) [53], a standoff XML format designed to represent a wide range of linguistically annotated textual and multimodal corpora. It uses a layer-based architecture in which each layer can contain another linguistic annotation, such as part-of-speech annotations, lemmatisations, syntax trees, coreference annotation etc., each stored in separate XML files but refering to the same raw data. To cope with the different descriptive schemes of emotion a XML-based language named EARL[3] (Emotion Annotation and Representation Language) has been developed. EARL supports discrete and time-varying encoding of emotion dimensions as well as independent annotation of multiple modalities modelling of specific relations such as blending or masking of emotions [81]. Likewise, EmotionML[4] or short EML (Emotion Markup Language) [286] is another standard with the aim to strike a balance between practical applicability and scientific well-foundedness defined by W3C. It is not only meant for the purpose of annotation, but also automatic recognition and generation of emotion-related states.

Nevertheless, many fundamental questions are still to be answered. For instance, there is no common agreement on how many raters should be employed to judge the material. Some researchers demand that benchmarking should be done by at least ten evaluators [89], in reality it is often less. And should the raters be presented with a combination of all available data channels? Or with individual streams in a context free manner? While one may argue that more information will result in a higher agreement between annotators, some studies prove the exact opposite [81]. At least, nowadays researchers can draw on a number of annotation tools that have been released to the community free of charge. Popular layer based tools are ELAN (EUDICO Linguistic Annotator) [302, 345] (see Figure 4.1), ANVIL [172, 173], or EXMARaLDA (Extensible Markup Language for Discourse Annotation) [283] which offer freely definable tracks to

[3] http://emotion-research.net/projects/humaine/earl
[4] http://www.w3.org/TR/emotionml/

Figure 4.1: Classic annotation software like ELAN (left) allows describing user behaviour through time-anchored segments. More recently tools like GTrace (right) have been developed which provide mechanisms for annotating continuous attributes. Although the latter method is more time consuming and requires a higher amount of attention it allows for a more precise description of interaction dynamic.

insert time-anchored labelled segments. These tools are well suited to describe social cues that feature precise starting and ending times, such as speech and gestures turns. They also allow capturing longer states as long as the behavioural content can be decomposed into more or less homogenous segments. This, however, applies mainly to acted data, but can be rarely observed in everyday interaction. For instance, if one loses interest in a conservation it is unlikely this will happen in the blink of an eye, but is rather the consequence of a longer period during which he or she gets increasingly bored. The same applies to phenomenons like engagement or emotional feelings which – unless they are a reaction of a sudden and unexpected event – continuously evolve through time [213]. This, of course, makes it difficult or even impossible to fit them into rigid temporal intervals. Naturally this also has consequences for the recognition of those states. If it is not possible to manually tag behaviour with discrete labels, it is pointless to demand it from a machine. Instead, a continuous rating is required which at a regular time basis updates the current state on a dynamic scale (e.g. in an interval between 0 and 1) allowing for a more precise description of interaction dynamics. To this end, researches have come up with tools for annotating continuous attributes such as the intensity of a smile or the level of a negative emotion. Examples are Feeltrace [285] and its successor GTrace (General Trace program), which allow an observer to track the emotional content of an audio-visual stimulus over time based on activation-evaluation space (see Figure 4.1), or CMS (Continuous Measurement System) [211, 212], a tool for continuous expert coding of videotaped behaviour.

Although available annotation tools allow describing audiovisual material to a literally arbitrary level of detail, they still offer little automation. Hence, creating descriptions for several hours of interaction remains an extremely time consuming task. As a matter of fact, so far building large and high-quality data collections of human behaviour required a great deal of funding, which is the reason that most of the relevant datasets are results of long-term research projects, e.g. the European-funded HUMAINE[5] and AMI[6] project. However, since benchmarking efforts and

[5] http://emotion-research.net
[6] http://www.amiproject.org/

annotated databases are of prime importance to the social signal community [89, 267, 323] it will be important to ease the creation of large multimodal databases wherever possible, both in terms of recording and annotation. Unfortunately, the community is still lacking generic and well established open source tools to tackle the mentioned problems, in particular when it comes to recording setups that go beyond the standard setting of a single camera paired with a microphone. Since more data naturally requires more annotation effort at some point manual labeling alone becomes infeasible. Especially, if we think of the large quantities of data that can be automatically collected from the web, but which miss adequate descriptions. Therefore it will be important to reduce the costly effects of human rating by applying semi-automatic annotation strategies. One such approach is *Active Learning* [300]: only the samples most informative for the learning process are identified and presented to the human labeller. *Semi-Supervised Learning* techniques take it even further by completely refraining from human intervention [363]. Several strategies are found in literature which combine both strategies [192, 309, 314]. A novel method by Zhang *et al.* [362] called *Cooperative Learning*, starts by selecting unlabelled instances with medium confidence values, which it presents to human labellers. Afterwards, instances with high confidence are selected and the machine continues with the annotation. In their tests the authors could show that their approach outperforms several conventional approaches in both, performance and robustness. Crowdsourcing is another way to obtain high-quality data inexpensively and rapidly. One example is Amazon's Mechanical Turk [36], a web service that allows posting jobs (e.g. labelling a large number of photographs) which – for a certain payment – are then completed by people all over of the world. Hantke and colleagues [141] have developed a web-based multiplayer game which encourages player to label or record data by rewarding them with scores and prizes. The better the agreement with other players the higher is the score.

4.1.6 Lessons

The availability of new sensor devices like 3D cameras or wearables provides researchers with novel signals of great potential. Unfortunately, the use of such devices is often hampered as a higher level of expertise is required and no standardised ways exist to sync them with other equipment. Taking this into account, a core function of a system meant to support the development of social signal applications should be to provide a backbone for building up **complex recording setups including multiple and exotic sensor devices**. This requires uniform treatment of the captured sensor data, which has to be independent of the measured quantity, yet taking into account temporal characteristics.

Despite the variations in dimension and update rate, it must be guaranteed that captured signals share a common time-line. That is, at any point in time we want to be able to match the diverse channels and decide which measurement in one stream corresponds to a measurement in another stream. To this end, strategies will be needed to **keep captured signals in sync without relying on additional synchronisation hardware**. If necessary, a setup may include several machines which are connected in a network and have to be kept in sync as well. The stored data should be made available in **formats that are easy to access and widely supported** to ease further processing.

In addition to the raw sensor streams, automated creation of supplementary descriptions should be considered to bolster following annotation steps. This concerns the **pre-segmentation of the recorded material into regions of interest** as well as the **extraction of meaningful features and transcriptions**. So far, such treatments have been usually shifted into a postprocessing phase. However, extracting meta information on-the-fly opens up richer possibilities to steer experiments. A voice activity detection, for instance, could be used to control turn taking with a virtual agent in a Wizard-of-Oz like scenario without depending on human supervision. Hence, a recording setup should offer the possibility to enrich channels with additional processing steps to extract higher level information and control the visualisation of stimuli input.

4.2 Exploring Sophisticated Fusion Methods

In 1976 Harry McGurk and his research assistant John MacDonald wanted to study how infants perceive language at different developmental stages. Therefore, they had dubbed a video showing lip movements of a young woman pronouncing *ga* with the sound of the syllable *ba*. Surprisingly, when played back to subjects they would report to hear neither *ga* nor *ba*, but *da*. Since the discovery of this perceptual phenomenon, which is named after its discoverer *McGurk-MacDonald effect* or shortly *McGurk effect*, it has been thoroughly explored and led to a fundamental change in our understanding of how we perceive speech and language [128]. The revolutionary implication of this particular finding was that speech perception, which until then had been regarded as a purely auditory process, is actually influenced by vision. Basically, what happens is that our brain subconsciously pairs the auditory component of a syllable with its visual impression (received by lip movements). Now, if the impressions do no fit each other, sometimes the result is the perception of a third sound that lies somewhere between the two. This, of course, provides some strong evidence that in the human brain audiovisual information is processed mutually rather than in isolation. It is of particular interest that the integration of audiovisual seems to happen at a very early level, even beyond our conscious perception. In fact, the described effect still applies even if people are aware of it, i.e. there is no way to force our brain to ignore it. This suggests that the fusion of audiovisual information just like other sensory information (touch, taste, etc.), is an integral function of the human brain. In particular, this enables our brain (like that of other mammals) to always come up with a plausible explanation even in case of conflicting (or missing!) input. It is this wonderful ability that allows us with playful easiness to always form a coherent hypothesis about the current state of the world or – to stay in context – about the needs and goals of our interaction partners. And obviously if we want to build socially intelligent machines that perceive the world in a human like manner, we must not regard multimodal data as separated channels, but have to exploit manifold ways of fusing them similar to the human brain.

4.2.1 Early Applications

The integration of multiple data sources into a homogenous and consistent representation, which hopefully gives a more complete picture of the world as the independent sources would do, actually has a long history. Hope is therefore justified that a sensor, in effect, is constrained to measure a certain physical phenomenon over time. Hence, the desired information may be decomposed into incomplete (possibly overlapping) fragments with each sensor representing another piece of the cake. The task of any fusion system is therefore to collect the available pieces and – as far as possible – restore the original information from them [69]. Among the earliest applications of multimodal fusion have been military surveillance systems [311], which basically

boil down to the task of achieving an optimal detection and tracking of targets using multiple sensor sources. For instance, combining the output of a pulsed radar, which measures the range of a target, and an infrared imaging sensor, which determines the target's angular direction, allows a better localisation than either of the two sensors could provide. Other exemplary applications are guidance and control of autonomous vehicles, medical diagnosis, smart building, etc. [137].

The probably most popular algorithm to combine multiple measurements over time to estimate the unknown underlying system state is the so called *Kalman filter* [164]. Briefly spoken, a Kalman Filter updates the underlying system state by averaging the currently estimated system state, which is calculated based on previous estimations, with the new measurements of the sensors. The result is a new prediction of the system state, which lies between the previously estimated state and the new measurements. The purpose of the weighting between predictions and measurements is that sensors with a smaller estimated uncertainty can be "trusted" more and vice versa. This makes the Kalman filter especially useful in the presence of noise and malfunctioning hardware. Since the algorithm only relies on the previously calculated state and its uncertainty matrix, it is well suited for real-time applications. And it was this very feature that helped the Kalman filter to its greatest success: in the 1960s it was applied to navigation for the Apollo Project [129]. Today, the Kalman filter is one of several statistically based fusion approaches which try to predict an optimal estimate of the underlying system state by smartly fusing measurements of several modalities [215].

4.2.2 Basic Levels of Integration

Multimodal fusion always starts with the question at what level the integration is going to happen. Although we will learn about some special cases and variations, in principal any fusion approach can be categorised into one out of the following triple: *data level*, *feature level*, and *decision level* [69]. The choice depends on the properties of the streams to be fused, but also the general objective, e.g. whether the aim is to obtain a more complete picture, tweak prediction accuracy, or to increase the robustness of the system in the first place.

Data level fusion as an *early fusion* approach takes place at a very early stage in the processing pipeline. Since data is fused at a raw state, data fusion is only applicable if both input streams are of a similar kind. Fusion of infrared and visible images for illumination-invariant face recognition [142, 178] are an example. Beside its limited applicability due to registration and resolution match problems, data level fusion also suffers from sensor failure. *Feature level fusion*, which is also an early fusion approach, is the next higher level of integration taking place at an intermediate level somewhere between the raw sensor stream and higher-level decisions. Although feature level fusion still is not able to cope with incomplete information in one of the channels, it generally provides more flexibility in comparison to data level fusion. This is because individual sensor streams are first transformed into a common feature space. The feature vectors of all

modalities are then concatenated into a super vector. In this way, data from literally any type of sensor can be combined, e.g. features calculated from a video image can be fused with features extracted from an audio waveform.

Decision level fusion, also referred to as *late fusion*, integrates information at an even higher step, that is after applying some sort of classification to each modality. In the following decision fusion process only the probabilities from the various channels are taken into account [253]. This can be achieved by applying simple rule based strategies, e.g. by summing up the individual decisions or deciding in favour of the modality with the highest score. Or a second stage classifier is applied on top of the single decisions, which is sometimes denoted as *model level fusion* [225]. Generally, we can regard decision level fusion as being better resistant to individual sensor failures compared to early fusion strategies. In absence of a sensor it is still possible to draw on the decisions of the remaining modalities. The flip side, however, is that fusion potential lost at data and feature levels cannot be regained at decision level [69].

So, which of the triple can then be considered as the ideal level for integrating social cues and should therefore be the preferred choice for a given problem? As we will soon see, this simple question is the Holy Grail of multimodal fusion and no definite answer can be given yet.

4.2.3 Conventional Fusion Approaches

In the first place, we may ask what benefits we can expect from a fusion system and if it will actually bring an advantage over the single modalities. If we think in terms of the earlier example of a cake, which is cut in pieces and every sensor is delivering some part, multimodal fusion seems to pay off if different aspects of the observed social behaviour are carried over through different modalities. In the context of emotions, for instance, some emotions are more likely to be observed from the face, whereas others are mainly expressed through speech. And precisely that was found by De Silva *et al.* [72, 73] when they showed clips of six basic emotions (angry, happiness, sad, surprise, dislike, and fear) to 18 subjects in three conditions: using audio information only (whereby speech was incomprehensible to the subjects), using video information only, or presenting the original audiovisual recordings. Looking at the percentage of correct guesses they concluded that subjects recognised anger, dislike, happy and surprise better from the video channel, while sadness and fear were better recognised from the audio channel. But does this mean that fusion of both modalities will help to improve the automatic classification of emotional states? A straightforward way to figure it out is by comparing classification accuracy of the individual classifiers with that of a fused recognition.

Based on their finding, De Silva and colleagues suggested a rule-based method which would always decide in favour of the more dominant mode in the case that audio and video predict different classes [72]. After applying this relatively simple rule in a fully automated recognition

system, accuracy rates in the fused case were indeed higher compared to unimodal classification [52, 71]. In response to this encouraging outcome, further combination strategies were investigated in the following years adopting a similar kind of methodology. First, multimodal data is recorded in synchronous manner. Afterwards, experts review the data, marking segments of interest and providing them with labels. A single annotation is then applied to all modalities and features are extracted separately for each channel and either concatenated in case of feature fusion or fed into individual classifiers, and decisions are combined afterwards.

Busso *et al.* [39], for instance, concluded that feature-level fusion was most suitable for differentiating anger and neutral state while decision-level fusion performed better for happiness and sadness. For the classification of infant cries, Pal and his group [235] translated uncertainty derived from the confusion matrices of the single classifiers into belief vectors which they used to take a final decision by maximisation of the belief score. Sebe and colleagues [298] suggested the use of dynamic Bayesian Networks to model the interdependencies between audio and video data and handle imperfect data by probabilistic inference. Kanluan *et al.* [165] applied decision level by a weighted linear combination to classify emotions along three continuous-valued primitives, namely valence, activation, and dominance. Petridis and Pantic [245] built a system to distinguish laughter and speech episodes in audiovisual recordings using decision and feature level fusion. Aran and Perez [8] combined audio and visual nonverbal cues for dominance estimation in small group conversations at both, feature extraction level and at classifier level via score and rank level fusion. Similar, Hung *et al.* performed automatic estimation of most dominant person in a group meeting [149] as well as estimation of cohesion in small groups [148] from a combined vector of audiovisual cues. On a database of aggressive, cheerful, intoxicated, nervous, neutral, and tired behaviour in an airplane situation, Schuller and colleagues [287] mapped audio and video features into a common feature space, which allowed for overall feature space optimisation. Finally, various types of multimodal correlation models are based on extended artificial neural networks (ANN), e.g. [42, 116].

While the mentioned works exclusively focus on audiovisual fusion, studies in psychology suggest that combined face and body approaches also have great potential for the analysis of human expressive behaviour [5]. Based on facial expressions and body gestures, Gunes and Piccardi investigated bimodal fusion methods at different levels for emotion recognition [134, 135]. Kaliouby and Robinson [100] proposed a vision-based computational model to infer acted mental states from head movements and facial expressions. Castellano *et al.* [46] presented a multimodal approach for the recognition of eight emotions that integrates information from facial expressions, body gestures, and speech. Kim [170] explored ways of integrating speech and physiological measures for emotion recognition based on a short-term observation by means of feature level fusion and decision level fusion as well as a *hybrid fusion* scheme, which combines outcomes of the other two approaches. Karpouzis and colleagues [166] performed facial, vocal, and bodily expression analysis via a Recurrent Neural Network (RNN). Castellano *et al.* [46] de-

veloped a multimodal framework for analysis and recognition of emotion from expressive faces, gestures and speech, which they trained and tested with a Bayesian classifier.

Fusion systems that consider more than one level are rather the exception. Glodek *et al.* [125, 126] describe fusion architectures that apply fusion gradually on different levels, including fusion steps from levels of signal recognition to abstract logical inferences and therefore describe the three categories of perception-level fusion, knowledge based fusion and application level fusion. In [126] they define several requirements, e.g. compensation of sensor failures (missing data), consideration of the temporal dimension and applicability to open world settings. We will come back to these issues in Section 4.2.8.

Drawing a comparison between the presented works, two interesting observations can be made. For one thing, many of studies report results that clearly emphasise the usefulness of combined classification, often yielding improvements of 10% and more compared to the best unimodal prediction (e.g. [165]). At the same time, other studies exist which report no or only marginal improvements (e.g. [155]). For another thing, it appears to be extremely difficult to derive general guidelines in terms of an ideal integration level and fusion method. Some studies favour early fusion approaches (e.g. [245]), others find late fusion to be superior (e.g. [8]). So, what are the reasons for the unsteady performance of the proposed fusion systems and why is it so difficult to generalise between them? One could leave it to Busso [39] saying the best fusion method simply depends on the application. But while the latter is surely an important point, we must not forget that the performance of a fusion system also depends on the methodology that is used to evaluate it.

A meta study on 30 published studies on multimodal affect detection by D'Mello and Kory [78] comes to the interesting conclusion that performance improvement, i.e. the improvement of the fused decisions compared to the best unimodal classification, correlates significantly with the naturalness of the underlying corpus. While an overall mean multimodal effect of 8.12% is reported, they also found that improvements are three times lower when classifiers are trained on natural or semi-natural data (4.39%) compared to acted data (12.1%). At a first glance, this suggests that under realistic conditions there is less room for improvements than in case of acted material. However, especially in natural interaction people are supposed to make use of a mix of behavioural cues through multiple channels, which actually makes multimodal fusion even more promising as it calls for smart ways to assemble the incoming cues to a full picture. This suggests that conventional fusion methods developed in the context of acted data cannot be applied to more realistic settings without further ado.

Figure 4.2: Visualization of predictions for the first third of the DaFEx samples. White squares represent correct predictions and black squares failures [197].

4.2.4 No Free Lunch

In [197] we took this thought further. Originally, the work aimed to answer the question whether more sophisticated fusions algorithms can be expected to gain higher a performance compared to standard fusion methods, such as a simple *Product Rule*[7]. In a large scale study we carried out a systematic comparison of a total of 16 fusion methods on two affective corpora: the DaFEx database [20] as a classic acted corpus containing recordings of professional Italian actors. And the CALLAS expressivity corpus [41] featuring expressions by participants with no special acting abilities. Both databases were then processed in the same way. Comparing the best fusion result with the best unimodal channel, on the DaFEx data an enhancement of 8% was observed, while no improvement could be achieved on the CALLAS data. At least, results for the weaker modality were always exceeded by 8-10%. With respect to the tested fusion algorithms, standard fusion techniques performed on a very stable basis. More elaborate strategies such as the *Cascading Specialist* method [171] were among the best fusion approaches for the CALLAS data, but failed on the DaFEx data. All in all, we observed rather small differences in accuracy among all considered fusion techniques.

In search for an explanation why there is so little difference in recognition performance for the tested fusion strategies, we introduced a novel visualisation technique which allowed us to compare the massive number of results in a clear way. As shown in Figure 4.2 for the DaFEx corpus each tested sample is represented as a square in a vector, which is either filled white if the sample was correctly classified, or black otherwise. By row-wise aligning results from the different fusion methods we can immediately see for each sample which methods have output a correct prediction and which have failed. It stands out that many columns are either uniformly white or

[7]The *Product Rule* combines decisions by multiplying individual class probabilities.

black. This means fusion methods are likely to keep a decision if the individual classifications are unisono (two white or two black squares in the first two rows). Unfortunately, this is also true for incorrect predictions. Hence, we cannot expect improvements on samples were single classification fails. But what about cases were only one single modality was incorrect? Here, we often observe a random pattern, i.e. the fusion methods either follow the correct modality or are misled. Since none of the investigated fusion methods is able to reliably predict the right modality, they all end up with a more or less equal performance. This is what Wolpert and Macready denote as *no free lunch* theorem [349]: over all samples performance in average stays the same.

4.2.5 Contradictory Cues

Intuitively, we would say that integrating information from several sources should generally lead to an increase in recognition performance. But while this clearly holds for acted data, it does not seem to count for spontaneous data, at least not to the same degree. The crucial point here is that adding more information also increases the choices, especially since conventional fusion algorithms are designed to incorporate information from all modalities at all time. De facto this means that in case of contradictory signs they have to decide in favour of one, which can either turn out to be the winning ticket or a loss. This leads us to a more general question: to what extent do the social and affective cues, which are expressed across different channels, fit each other?

As long as acted data is the benchmark, it is in fact relatively likely that expressions in the various channels are in unison, which explains the solid gain in performance for convenient fusion methods. However, things turn out to be different when it comes to real-life observations. As discussed earlier, it has been observed that realistic scenarios make it more probable that users draw on a mixture of strategies to express emotion [17] leading to a complementary rather than consistent display of social behaviour [356]. In fact, large parts of naturalistic emotion databases have been proven to carry contradictory multimodal cues [81]. To find out if this might serve as an explanation for the poor fusion performance on the CALLAS corpus, we decided to create separate annotation tracks for each channel and compare them afterwards [337].

The annotation scheme that we used to categorise the samples in the CALLAS data is based on the valence-arousal space introduced in Section 2.4.2. Each quadrant is mapped to a discrete class: *negative-low*, *negative-high*, *positive-low*, and *positive-high*. Since we deal with an audio-visual corpus, common practice would be to provide an annotator with the full material. But in this case it remains concealed which sense an annotator has actually relied on: rather what he saw or what he heard. Hence, we decided to do two runs. In the first run, we asked three experts to independently label samples exclusively on the audio channel. Several weeks later the procedure was repeated by the same raters, but this time on basis of visual information only. Final combination was done via majority decision, as three assessments are given to each orientation of valence

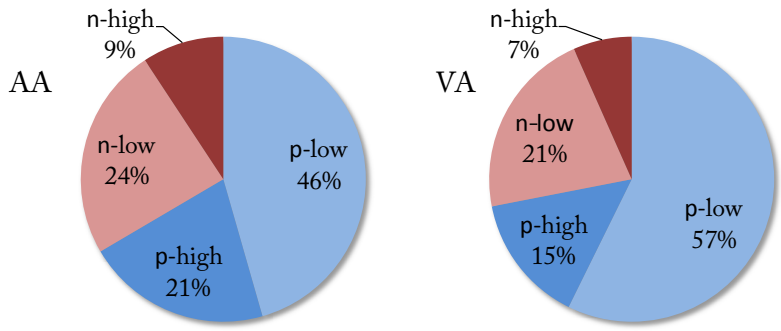


Figure 4.3: CALLAS corpus [337]: differences in the distributions of the class labels suggest that probands are more expressive through the audio channel. For instance, in the audio based annotation class plow occurs with a 10% smaller frequency compared to the video based annotation. AA=audio based annotation, VA=video based annotation.

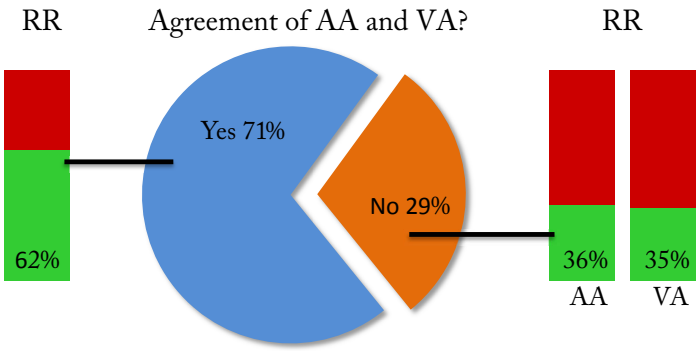


Figure 4.4: CALLAS corpus [337]: amount of correctly classified samples is significantly higher for samples where both annotations agree. AA=audio based annotation, VA=video based annotation, RR=recognition rate.

or arousal respectively, definite decisions were found. Measuring overall inter-rater reliability in terms of Fleiss' *Kappa value* [115] indicated a moderate agreement of 0.52[8]. However, looking at the Kappa value for valence and arousal independently revealed that agreement for valence was much higher (0.84 and 0.71, which implies high agreement) compared to arousal (0.38 and 0.48, which implies only a fair agreement). Apart from this, we also observed a higher agreement for valence if annotation was based on audio recordings, while the agreement for arousal was higher if based on video recordings. A clear sign that emotions in the studied corpus are not necessarily expressed in a coherent way across modalities.

This becomes even more obvious when comparing the distributions of the class labels between the two conditions, which is done in Figure 4.3. From the graphs we see, for instance, that *positive-low* has been assigned with a 10% higher frequency in video-based annotation. Overall, almost one third of the samples in the observed corpus were assigned a different class label be-

[8]Fleiss' Kappa is expressed as a number between 0 and 1, where 1 indicates a perfect agreement.

tween channels. It appears only natural that these samples are preferred candidates for mistakes if fused by a classifier. Indeed the amount of correctly classified samples is significantly higher when both annotations agree (see Figure 4.4).

4.2.6 Consequences

Now, what are the consequences for a fusion system? Can we blame an algorithm if it favours another modality than the raters? After all, is it a good idea to force a decision by incorporating information of all channels at any time?

Let us assume we measure emotion from two channels and observe a positive state in the first channel and a neutral in the second. What answer should be given by the system? Probably we would expect it to output positive. But what if it gives neutral? Is this a completely wrong answer or maybe partly right? This becomes even more tricky if we use the sample to train the system. Should we assign a positive or a neutral label? No matter how we decide we will inevitably introduce errors in the other channel. At least as long as we assign a single class label to each sample.

As mentioned before, conventional fusion approaches start from a single segmentation, which is applied to all modalities. The weak point of this approach is that this way for each segment a decision will be forced from all modalities. In case of audiovisual emotion recognition, for instance, it is a common strategy to trigger analysis by the audio channel with the consequence that facial expressions are only considered when voice is active [197]. This approach of triggering multimodal fusion from a single annotation or modality has at least one severe drawback: additional cues in further modalities can be expected but are not guaranteed. Important facial components, e.g. may be signalled before the start of an utterance and in the worst case may already be back to neutral at that point. It may even be the case that lip movements during speech production distort the facial expression. A neutral or misleading facial expressions during the utterance may then have a negative influence on the fused decision (see Figure 4.5).

We can assume that the described situation is more likely to occur in realistic scenarios where users show rather subtle than exaggerated expressions, which would explain why the performance gain of conventional segmentation based fusion methods in that case is significantly lower than for acted data (see Section 4.2.3). Reasons for observing complementary or even contradictory cues are manifold. As just mentioned it can be simply a matter of anatomic restrictions if e.g. somebody is speaking and hence is less expressive around the mouth. It may, however, also be the case that a user tries to hide his emotions and therefore speaks in a neutral voice, while yet his face gives off signs about his actual affective state [94]. Or because someone is generally more expressive in one modality than the other. For instance, there is evidence that valence is rather expressed through the face, whereas signs of arousal are mainly found in the voice [110].

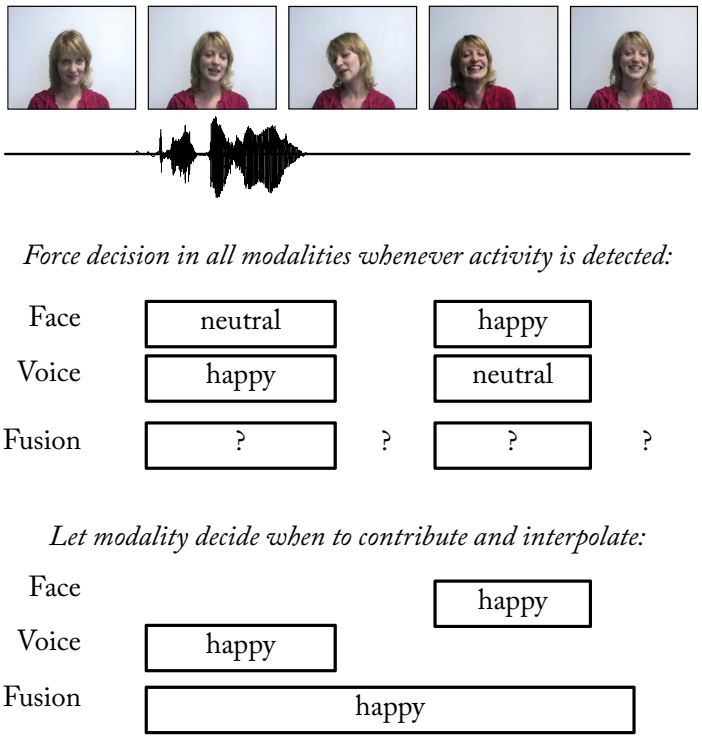


Figure 4.5: In conventional fusion approaches information is combined over fixed time segments, e.g. between beginning and ending of an utterance. This has the drawback that cues from other modalities outside the segment will be missed. In the shown example the lady starts talking with a positive undertone, while her face still shows a neutral expression. Afterwards, when a smile is detected the audio channels remains neutral. This leaves the fusion algorithm whether to trust the face or the voice. This also applies to periods without activity. To overcome these limitations, the lower part of the figure sketches an alternate fusion approach which combines cues asynchronously: Instead of postponing decisions until activity is detected and then forcing all modalities to contribute modalities can contribute individually. To fill the gaps between contributions the fusion system could apply some sort of interpolation.

4.2.7 Asynchronous Fusion Strategies

So are there ways of combining modalities without forcing decisions from all channels in every time slot? For a corpus consisting of speech and physiological signals, Kim *et al.* [170] suggested choosing the borders of the single segments in such a way that it lies in the middle between two spoken utterances. In this way physiological cues before and after an utterance, and not only during an utterance, are taken into account. Although decision taking is still triggered by the audio channel, it is a step towards relaxing the requirement of a strict synchronisation of the channels[9]. A more sophisticated way to tackle the described problem is offered by dynamic classification. Since dynamic classifiers work on continuous streams of short-term features, it is not necessary to force a fusion decision "from above". Instead, they (principally) have the ability to model

[9]To avoid confusion, it should be noted that the streams themselves still have to be in sync. It rather means that the fused information from the single channels may not necessarily represent the same time interval.

temporal relations between the streams and learn when and how multimodal information should be combined. To this end, Song *et al.* [304] proposed a tripled Hidden Markov Model (THMM) which is able to integrate three or more streams of data and allows the state asynchrony of the sequences while preserving their natural correlation over time. Zeng *et al.* [357–359] applied Multi-stream Fused Hidden Markov Model (MFHMM), where state transitions of different component HMMs do not necessarily occur at the same time across different streams so that the synchrony constraint among different streams is also relaxed. Coupled Hidden Markov Models (CHMM), where the probability of the next state of a sequence depends on the current state of all HMMs and therefore enables an improved modelling of intrinsic temporal correlations between multiple modalities, have also been proposed (see Nicolaou *et al.* [225]).

To overcome the computational complexity of asynchronous Hidden Markov model (AHMM), Wöllmer *et al.* [347] suggested a multidimensional dynamic time warping (DTW) algorithm for hybrid fusion of asynchronous data, requiring significantly less decoding time while providing the same data fusion flexibility as the AHMM. Finally, Artificial Neural Networks (ANN) offer a third alternative for asynchronous fusion; in particular in the form of Long Short-Term Memory Neural Networks (LSTM-NNs), which replace the traditional neural network nodes with memory cells, essentially allowing the network to learn when to store or relate to bimodal information over long periods of time. In fact, LSTM-NNs have been successfully applied to combine acoustic and linguistic features to continuously predict the current quadrant in a two-dimensional emotional space spanned by the dimensions valence and activation [348]. Likewise, in a similar emotion recognition task, this approach successfully fuses facial expressions, shoulder gestures and audio cues [226].

While the aforementioned fusion approaches outperform segmentation based schemes, one severe disadvantage comes from their complexity in terms of training and decision taking. Since it is difficult to understand how the network reaches a decision, applying it in a real-time system bears the risk that the learned model parameters may poorly translate if applied in a possibly less controllable environment. However, once trained, they function as a black box whose hard-wired parameters leave little opportunity for adjustments to the new conditions. Transfer learning defines a possible solution [236], for instance by mapping features in the source domain to the target domain using an additional neural network [75].

4.2.8 Event-driven Fusion

In an early paper on audiovisual fusion by Chen and Hung [51] in 2000, the authors note that bimodal processing only makes sense when the user is speaking and the camera is successfully tracking the face so that facial features can be reliably extracted. They also suggest to use mainly audio features during speech events and combine them with the facial expressions before and after the sentence, when pure facial expressions are likely to occur. This actually raises another

interesting and often overlooked issue: most algorithms start from the assumption that data from all modalities is available at all time. Of course, in offline mode this condition is easy to meet by simply omitting parts where input from one or more modalities is corrupted or completely missing. Thinking of a real-time system, on the other side, obviously this assumption no longer holds. Either, as remarked by Chen and Hung [51], because no useful information *can* be detected (e.g. the user is not looking into the camera), or because there *is* nothing useful to detect (e.g. the user is not talking), or last but not least, due to a failure in one of the sensors. In short, to be used in a realistic application a fusion system has to be able to handle *missing data* in any modality at any time. We will address this issue again in a later Section (4.3.1).

One way to tackle the problem of missing data and making the fusion process more transparent is by shifting from segmentation based processing to an *event-driven* approach. Introducing events as an abstract intermediate layer effectively decouples unimodal processing from the final decision making. In this view, each modality serves as a client which individually decides when to add information. Signal processing components can be added or replaced without having to touch the actual fusion system and missing input from one of the modalities does not cause the collapse of the whole fusion process. In some sense, this kind of event-driven fusion is similar to semantic fusion used to analyse the semantics of multimodal commands and typically investigates the combination of gestures and speech in new-generation multimodal user interfaces (see the following section). However, only few attempts have been made to apply event-driven concepts for automated emotion detection.

In an artistic Augmented Reality installation, the Callas Emotional Tree [123], Gilroy *et al.* use event-driven fusion to derive the affective state of a user in real-time. The basic idea of their approach was to derive emotional information from different modality-specific sensors and map it onto a continuous affective space spanned by the three dimensions Pleasure, Arousal and Dominance (PAD model) [124]. Since the application depended on a continuous assessment of the affective user state, the current state of the fusion system was constantly represented by a vector in the PAD space. And the direction into which the vector would move was set by a bunch of vectors representing the single modality-specific contributions. The values of those guiding vectors was updated whenever a new affective cue was detected or otherwise decayed over time. A different approach for predicting user affect in a continuous dimensional space based on verbal and non-verbal behavioural events (e.g. smiles, head shakes, or laughter), has been published by Eyben *et al.* [108]. In their system events are seen as "words" which are joined for each time segment and converted to a feature vector representation through a binary bag-of-words (BOW) approach. Tests on an audiovisual database proved the proposed string-based fusion to be superior over conventional feature-level modelling.

4.2.9 Semantic Fusion

While in the field of social signal processing event-based fusion is still the exception, another area of applications related to multimodal information bases almost completely on events: *semantic fusion*. Semantic fusion investigates the combination of natural and intuitive interaction styles such as gestures and speech for developing new multimodal interfaces that go beyond traditional interaction with keyboard and mouse, also referred to as WIMP (windows, icons, menus, pointer) interaction, to support and accommodate a user's perceptual and communicative capabilities. It is achieved following a *synergistic* strategy [229] which allows processing multiple input modes in parallel and model their temporal and semantic combinations [146]. For instance, a user may point to a spot on a street map and express the wish to navigate there. Seen in isolation the cues in the gesture and speech channel do not contain enough information for a correct interpretation. Hence, the objective of semantic fusion is to achieve the mutual disambiguation of the fragmental information provided by different modalities [205].

First attempts date back to the 1980s starting off with Bolt's famous "Put-That-There" system [31]. The interpretation engine of the system allowed users to interact with a set of virtual objects using both, speech and hand gestures, simultaneously. Users benefit from such kind of interaction since the same goal can be achieved in a various and thus often more intuitive ways. On the one hand, this may reduce the costs to learn the system as users can interact depending on their personal preference. On the other hand, it is likely to speed up interaction as users can always pick the strategy which is most convenient in a certain situation. In fact, studies in dynamic map tasks have proven that users show a strong preference to interact multimodally, rather than unimodally [233]. However, systems offering this kind of support for natural integration patterns also have to deal with the fact that spoken and gestural modes are more likely to supply complementary rather than redundant semantic information. In other words, situations with ambiguous user input are likely to occur and have to be dissolved in some way. For instance, if the user refers to an object with "this", the addressed object has to be automatically derived from other input channels, e.g. it has to find out if there is an accompanying pointing gesture. This is further complicated by timing issues as present information is often used or referred later, gaze e.g. tends to precede spoken commands [200].

Early approaches use a slot buffer to which they feed relevant input events and which is then compared with a set of possible facts defined in a command vocabulary. If one fires the according command is executed, otherwise the system waits for more input information [200, 234]. A more formally and reusable mechanism was suggested by Johnston *et al.* [158]. Their approach models integration by a *unification* operation over *typed feature structures*, which determines the consistency of partial information and, if appropriate, combines it into a single result. A feature structure is a set of attribute-value pairs, for example a tuple of an object and a location (which may both be again represented as a feature structure). Partial user input is modelled as

an underspecified feature structure, i.e. it contains features that are not instantiated. If input is received which fits an underspecified feature it can be integrated. For example, a pointing gesture may supply the missing location feature for a speech input describing an object. To enable continuous and incremental parsing, and provide better support for specify temporal, spatial and semantic constraints, alternate approaches have been proposed, including *unification-based multimodal grammars* [156], *finite-state automata* [157], *petri nets* [222], or *temporally augmented state-transition networks* [187]. Mehlmann *et al.* [205] present an approach that combines the advantages of previous unification-based, rule-based and finite-state-based approaches in a single system using declarative and visual modelling paradigms. A good overview of fusion engines for semantic fusion is given by Lalanne *et al.* [185]. An approach that applies a mixture of semantic and signal fusion have been proposed by Bosma and André [32]. To avoid false interpretation of especially short utterances, which tend to be highly ambiguous if only the linguistic part of the message is considered, they integrated the meanings of spoken input and combined it with the emotional state assessed by physiological data, such as heart rate. Crook *et al.* [64] tried to improve the robustness of a speech recogniser by fusing emotional states recognised from the acoustics of speech with sentiments extracted from the transcript of speech.

4.2.10 Lessons

Although it is commonly agreed that integrating information from multiple channels bears great potential, most studies still focus on a single modality. Works concerned with the fusion of multimodal data often dismiss the complex temporal relationships that exist between the diverse channels. Given that we are able to capture and process synchronised multimodal data, possibilities are needed to **combine information on different levels of processing**. To support fusion at an early state means including constructs that allow merging streams, either at a raw stream level or at feature level.

Early fusion requires the streams to be of a similar kind which is not always a suitable solution. Alternatively, fusion may take place at decision level, too. This, however, requires a representation of information apart from continuous signals. This is because detection results do not necessarily transform into another continuous signal. A gesture recogniser, for instance, could try to detect on- and offset of a gesture path and map it to a certain gesture class. The result is a sequence of independent decisions representing disjointed periods in time of variable length. To provide a suitable data structure for such situations an architecture is needed to **handle higher level information in form of events**. In contrast to continuous signals, events do not have to occur in regular intervals but may pop-up any moment for a certain time. Still, we want to be able to put them into temporal relation with other events as well as continuous streams.

As pointed out before, the majority of conventional decision fusion approaches assume that important social and affective cues occur simultaneously throughout all modalities. This, however, does not necessarily hold for spontaneous expressions. Representing recognition results as a loose list of discrete events makes **room for novel fusion strategies**. In particular, it offers the possibility to search multimodal cues in each channel by taking into account individual temporal phrasing. Afterwards, detected cues can be combined according to their chronological occurrence. Providing facilities for both, synchronous and asynchronous fusion approaches, provides a promising testbed for novel fusion approaches.

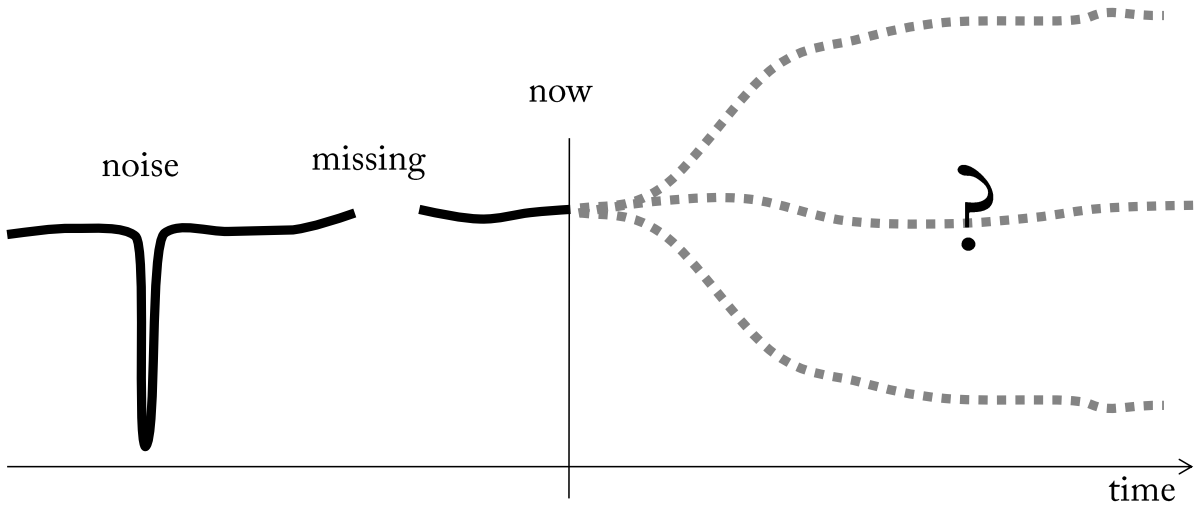


Figure 4.6: Building an online system creates its own challenges. In offline processing, for instance, signals are available as a whole from the very beginning, whereas in online processing we only know that part of a signal that has been processed so far. Since an online systems requires a continuous processing of the input streams, we need strategies to deal with noise and react to missing data.

4.3 Online Recognition

Systems developed of offline use, cannot be applied without further ado in real-life applications and the effort to transform them into real applications is easily underrated. In offline studies it is common practice to take a set of prerecorded files and process them in independent steps. Only when all files have successfully passed a stage, results are handed over to the next step. Typically, this starts by reviewing the data and excluding parts where things went wrong, e.g. due to sensor failure or a too noisy environment. More data might be removed to balance the number of samples per class or to remove parts with sparse interaction. In order to account for differences between subjects and between successive recording sessions, normalisation is applied over signals as a whole. In case of multimodal fusion, it is common practice to process the various channels separately and combine features or decisions afterwards. In an iterative manner each step is then fine-tuned until a good configuration is found and the best accuracy is reported as a sort of benchmark what can be achieved with the system. While this defines the common way to evaluate the performance of a recognition approach and offers a convenient format to compare results, it says little about the applicability in real applications. In fact, it may require a good deal of work to turn it into an online system, while it cannot be taken as granted whether it will still perform satisfactory. This is because an online algorithm does not have the entire input available from the start, nor can it leave out certain parts of the input (see Figure 4.6). Instead, all input needs to be processed in small portions in a serial fashion. The following section is dedicated to the manifold reasons which make it so challenging to build an online recognition system.

4.3.1 Missing Data

In an online setting we must cope with situations in which the observed signal is not usable at all, e.g. due to sensor failure. The issue of *missing data* is a phenomenon usually ignored in offline studies since at a first glance it does not make sense to keep parts of a signal that are corrupted or do not carry useful information. An online system, however, should come up with a solution to handle it, so that robustness of recognition performance can be guaranteed.

Generally we can identify various causes for missing data: a sensor device can fail so that an according signal is no longer available. Even if a sensor device is running properly there is the possibility that desired information within a signal is no longer accessible, e.g. a facial expression recogniser will have a hard time in detecting signs of emotion if there is no visible face in the video image. We can also think of a situation in which the desired information theoretically is at hand but practically corrupted to some extent, e.g. a speech signal that is overlaid by noise. Finally, not only technical problems can be responsible for one or more modalities to become useless. If a subject simply does not generate observable material, no meaningful data can be recorded, e.g. the gesture modality will not contribute relevant information while monitoring a momentarily motionless user. A system capable of handling missing data must therefore dynamically decide which channels are available and to what extent the present signals can be trusted. For the case that data is partially missing a couple of treatments have been suggested in the literature: Multiple imputation predicts missing values using existing values from previous samples [237]. In data marginalisation unreliable features are marginalised to reduce their effect during the decision process [74].

In a single channel classification problem, missing data means that no reliable output can be given, although it might still be possible to make a prediction based on previously seen data. In a multimodal scenario with several input nodes, however, things look different. If a single channel fails there is still a good chance to find useful information in the other channels. This also applies if one channel is interrupted by noise and hence can less be trusted. Once more, the tricky part is to dynamically decided in which channels to exploit in the fusion process and to what extent the present signals can be trusted. Techniques such as Kalman filtering [164] or adaptive fuzzy systems [62] are well known for their application in sensor fusion to reduce uncertainty and produce a more accurate prediction than any of the original signals considered separately [270]. Zeng [361] proposes a fusion method based on Multi-stream Fused Hidden Markov Model (MFHMM): if one component HMM fails due to some reason, the other HMM can still work. In [332] we have shown that many of the standard fusion techniques can be adjusted to successfully cope with missing data. A simple yet efficient strategy e.g. is to maintain single channel classifiers as a backup if the fusion model is not applicable.

4.3.2 Non-Prototypicality

Another property that distinguishes online from offline processing is the fact that a fully automated online system has to be prepared to handle incoming data without exception. Even the decision to skip samples has to be made in real-time based on the incoming streams. In case of emotional speech processing this has been referred to as an *open microphone* or more generally *open recording* setting [306]. Of course it is principally possible to demand this in an offline study, too, by include literally everything that was recorded, but in practice it is unlikely to find corpora which keep noisy and interrupted data, or including long parts during which nothing of interest happens. And since by definition a data set is dedicated to a specific sub-domain it is usually not meant to reflect real-life in all its facets. An emotional speech corpus, for instance, which includes utterances in all of Ekman's basic emotions misses blended and subtle nuances as they occur in natural interaction. Hence, the trained system will be tuned to detect *prototypical* emotions, but we cannot expect satisfactory performance for what Steidl *et al.* [306] call the *hinterland* of emotions. A fact experimentally proven in [219, 299]. Unfortunately, the hinterland is what we will mainly have to deal with in real-life applications.

Handling of *non-prototypical* behaviour has therefore claimed to be one of the most challenging barrier when moving to real-life technology [293]. Fortunately, this does not mean that a model is required which is able to precisely know and classify each and every possible state a user may adopt. Even a real-time application will still function within a specific context and must only detect behaviour relevant to its task. Yet, it should properly ignore anything that is not relevant and map it into a *garbage class*. Hence, an according garbage model has to be trained by including non-relevant data in the training phase. Something usually completely left out in offline studies as the designer is given the possibility to choose data according to his needs. In a real-life scenario, however, pre-selection is no longer feasible. In fact, it can be expected that a good deal of the processed input will be irrelevant. For instance, a recogniser trained to detect laughter bouts will not trigger many laughs during a serious conversation with your boss.

4.3.3 Continuous Recognition

Another promising approach to deal with different shades of expression is by shifting from discrete categories towards a continuous prediction. The idea is sketched in Figure 4.7. Basically, static categories are replaced by numerical values allowing for a better modelling of the often subtle and blended expressions observed in naturalistic settings. This, of course, requires according classification schemes that are able to interpolate between training samples and build a continuous decision space.

As a workaround some studies have used quantisation to map a continuous range onto discrete levels [175, 346]. An early step towards a true continuous dimensional affect prediction was

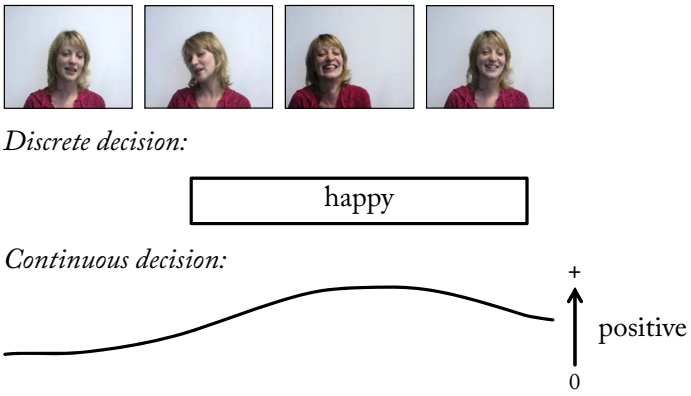


Figure 4.7: In continuous classification static states are replaced by dynamic values, which make room for a more fine-grade description and allows for a better modelling of the often subtle and blended expressions observed in naturalistic settings. The technique is especially powerful if dimensional descriptions are used, e.g. a label like "happy", which has a very specific meaning, can be replaced by a more general term like *positive*.

made by Hanjalic [140] who investigated the correlation of some basic video and audio features in affective movies with a valence-activation space. In the recent years a couple of studies have been published proposing approaches to realize continuous classification: Wöllmer *et al.* [348] and Nicolaou *et al.* [226] performed regression using a combination of Long Short-Term Memory and Dynamic Bayesian Networks (BLSTM-NN) classifiers. Wu *et al.* [351] tested three approaches (Robust Regression, Support Vector Regression (SVR), and Locally Linear Reconstruction) for emotion primitives estimation in a three dimensional space spanned by valence, activation, and dominance. Metallinuou *et al.* [214] applied a Gaussian Mixture Model-based approach to map a set of audio-visual cues to an underlying emotional state.

Although results look promising there is unfortunately a lack of adequate training corpora. Training a continuous classifier requires a ground truth of same format. In Section 4.1.5 tools such as GTrace and CMS have been introduced, which allow a continuous annotation of the reviewed material. Thanks to these tools few databases exist featuring continuous annotation, including the Belfast Naturalistic Database [79], the SAL corpus [83] and the SEMAINE corpus [202]. However, since the annotation process happens in real-time, using continuous descriptors is not only more time consuming, but also requires a higher amount of attention and cognition processing compared to discrete annotation task [213]. Furthermore, dimensional concepts, such as valence or dominance are less intuitive, which in combination with person-specific annotation delays generally leads to decreased inter-annotator correlations [213]. Yet, more effort is required to push research in this direction, probably supported by machine-aided transcriptions wherever applicable.

4.3.4 Segmentation and Incremental Recognition

Finding an appropriate unit of analysis is another issue which requires special attention when moving from offline towards online systems. In offline studies the length of a processed segment can be optimally chosen with respect to the boundaries of the observed user behaviour. In fact, it is common practice to align segments with the annotation track to make sure samples are as homogeneous as possible. Although it benefits the training process, it is not possible to apply the same kind of optimised segmentation in an online system for two reasons: First of all, we may not know in advance where the optimal boundaries will be. And second, a fluent interaction may require a fast prediction and there might be not enough time to wait until the optimal boundary is reached [293]. In fact, deciding when to start a segment already can be a tricky problem in itself. In speech analysis this is still fairly easy since periods of speech are usually surrounded by silence (given a clean signal). But in gesture analysis, for instance, the boundaries are fluid making it difficult to define when a meaningful sequence starts. A common solution to this problem is to manually trigger segmentation, e.g. by introducing a special start gesture or use something like *push-to-talk*. This, however, is only acceptable as an intermediate solution as it interrupts the natural interaction flow.

In the context of hand gesture recognition Alon *et al.* [4] propose a framework which detects multiple candidate hand locations of which a matching algorithm selects a single optimal sequence. Their approach reveals the various difficulties which need to be tackled to accomplish a truly automatic real-time detection system. Special attention is put to what is known as *sub-gesture problem*, i.e. some gestures may be very similar to sub-gestures of other gestures. Such relations are automatically learned by their algorithm and considered when choosing among competing gesture models. A pruning method is then applied to reject poor matches and decrease the space of possible matches. As soon as the pool of candidates reduces to a single match it is returned, i.e. it is not necessary to await the end of a gesture to recognise it, something known as *eager recognition*.

Humans are extremely good in coming up with an interpretation before all information is processed. This helps accomplish a fluent interaction and requires a so called *incremental processing*. For instance, when hearing a sentence we usually get its meaning before it is completed, which gives us time to come up with a proper response. Again in the context of gesture recognition, Kristensson and Denby [182] propose a system that estimates the posterior probabilities of the user's currently incomplete stroke to predict a user's intended template gesture. This enables them to provide continuous feedback to the user while producing a stroke. Similar approaches are needed in other areas, too, where a fluent interaction is wished that requires quick response to user behaviour.

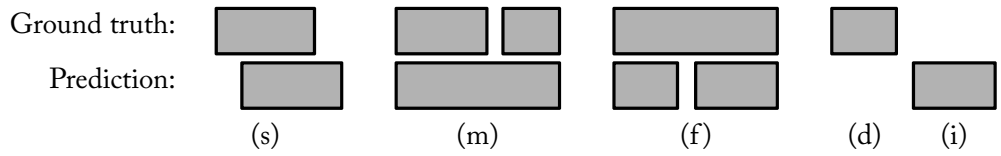


Figure 4.8: In online classification segments have to be found in an automatic manner. The figure shows typical errors: (s) a segment is shifted in time, (m) two segments are merged, (f) a single segment is fragmented, (d) a segment is completely missed, i.e. deleted, or (i) a new segment is inserted. During evaluation we have to consider that errors range widely in severity, e.g. a slightly shifted segment might still trigger correct system behaviour whereas an inserted segment could lead to a completely inadequate response.

4.3.5 Online Evaluation

In a classic offline study where segments are pre-defined it is straight forward to evaluate the quality of a classifier: just compare the actual class (*ground truth*) of each sample with the prediction and determine the percentage of correctly classified segments. To gain some better insights a *confusion matrix* can be constructed, which provides information which classes are frequently mistaken. Several other measures have been proposed to judge the performance of a classifier such as *specificity*, *precision* or *F-score* [255]. However, they are all based on the assumption that a sample is either correctly classified or not. As shown in Figure 4.8, this no longer holds for online classification where segmentation becomes part of the detection process, which introduces new sources of errors, such as fragmented or inserted segments.

One of the few studies which explicitly draw a comparison between offline and online evaluation has been published by Scherer *et al.* [279]. The system they describe spots laughter in naturalistic multiparty conversations from audiovisual data. The corpus consists of two natural and spontaneous conversations (90 minutes each) between four participants. Since the dialogues are not scripted or interrupted in any way (users were even allowed to move around freely) they resemble the input of an online system. Laughs and speech segments were manually annotated to obtain a ground truth for the following experiments. In the offline case speech and laughter segments were extracted from the annotations and treated as atomic units (parts with silence were not considered). Using Support Vector Machine (SVM) classifiers the offline system was able to spot laughs with an error of only 3.7%. In the online case, on the other hand, no data was removed and it was left to the classifier to automatically find appropriate segmentations. Therefore, online classification was performed over a sliding window with a length equal to the average length of the laughter and utterance labels of the offline data. The most obvious way to evaluate the outcome is via frame-by-frame comparison. However, since it is often the case that predictions do not precisely match the annotated segments this leads to a drop in performance of about 3%, although most of those errors are probably due to "fuzzy" boundaries. The authors therefore propose an alternate metric: To count as a hit it is sufficient if a laugh is detected somewhere within the bounds of an utterance and only if no spike is detected it is counted as a miss. Spikes

outside the bounds of a labelled laugh are defined as false alarms. The numbers are then used to calculate the F-score, which in the best case becomes 0.72. Interestingly, this was achieved using a Hidden Markov Model (HMM) approach, which in the online case outperformed SVMs, probably due to a better temporal modelling.

Finding smart ways of matching real-world events with recognised events has been intensively discussed in the area of *activity recognition*. In [338] Ward *et al.* propose a method that also takes into account when single events are *fragmented* into smaller events or several small events are detected as a single *merged* event. Although correct is only assigned when the prediction perfectly matches the ground truth, they distinguish different types of errors. For instance, if a ground truth event is only partly covered by prediction this defines an *underfill*, whereas prediction events that spill over the ground truth define an *overfill*. Errors are presented in a so-called *Segment Error Table* (SET) which summarises counts of the different error types giving a better insight what is going wrong. In a later publication [339] they extend their earlier approach by introducing a system of frame-by-frame metrics, which makes it possible to distinguish between frames that belong to a "serious" error. For instance, if a recognised event has an overlap with the ground through adjoining segments will be considered as *timing errors* instead of simply labelling them as *false positive*.

Finally, measures to evaluate discrete classes are also not directly applicable to dimensional approaches [133]. To measure the offset between ground truth and prediction Wöllmer *et al.* [346] propose the *Mean Squared Error* (MSE). It has the disadvantage of heavily weighting outliers, though [24]. As an alternate metric Grimm *et al.* [130, 132] propose the *Evaluator Weighted Estimator* which is able to capture linear structural patterns inhibited in the data [226].

4.3.6 Increased Implementation Effort

Apart from the fact that recognition performance gained in offline studies is usually overoptimistic, transforming an offline approach into a real-time capable system, can be tricky and sometimes even impossible. The crucial difference when moving from offline to online processing comes from the fact that a signal no longer can be accessed as a whole, but only in form of small data chunks as delivered by the sensor device. And in order to be fast enough to support interactivity, it is desirable to keep these chunks as small as possible. This, of course, introduces some fundamental limitations to the way in which signals can be processed:

- Since the data chunks that are delivered by the sensor do not necessarily fit processing length, some sort of buffering is required to store incoming data until the desired number of samples is reached.
- To guarantee continuous access to the sensor data and avoid deadlocks in later processing, subsequent processing steps should be executed in parallel.

- Apart from that, special treatment is required to ensure the synchronicity between modalities. If a channel updates faster than another it might be necessary to artificially delay the incoming data stream in order to combine temporally matching signal parts. In any case, a successive and separate processing of modalities as in offline systems is no longer feasible.

- The time it takes to process a data chunk in average must not exceed the time until the next frame is ready. Otherwise, even small latencies will sum up to a point where the system can no longer be considered to react in real-time. This makes a huge difference to offline systems, where time is (almost) no factor and the time it takes to process a frame may very well exceed the length of the frame.

- Since in offline analysis, each processing stage is delayed until the preceding step is finished, not only the raw, but also intermediate signals can be accessed as a whole, opening up sophisticated ways for artifact removal and normalisation based on global statistics. In an online system, where global statistics are not available until the end of a session, algorithms are bound to make predictions on what has been seen so far.

- Furthermore, an online system is forced to deal with any kind of input, i.e. even the decision to skip a part of a signal because it is overlayed by noise or disrupted in some way, has to be derived from the signal.

- Once running, it becomes difficult to assess a system's performance which would allow the system to tune itself to the current situation.

Hence, real-time capability, restriction to previously seen samples, and self-adaptation without having access to some sort of ground truth, define essential properties for any algorithm meant to be used in online applications. On top of this, buffering, parallel execution, and synchronisation add additional requirements, causing a notable increase in implementation effort. Yet, a proper transition is no warranty for success. Systems developed in offline studies tend to work only in (often highly) constrained environments [239] and even with an accuracy close to 100% under laboratory conditions a system may still perform unsatisfactorily in real-world scenarios [289]. Hence, additional adaptation might be necessary in order to increase the robustness and applicability in the online version.

In the end, it is probably the extra expense that explains why still most publications are confined to offline studies. The dilemma is that plain offline studies as they are often found in the literature convey a too optimistic picture of what can actually be achieved with a proposed system. This is because they tend to avoid problems which only occur if a system is tested in real environments. For instance, focusing data sets with few prototypical classes avoids the need to handle unwanted behaviour (*garbage*). Such non-prototypical behaviour defines a crucial problem in real life interaction, where a system has to deal with unexpected behaviour and input that is actually

Name	Language	Open Source	Modality	Online
iBUG Smile Detector	Matlab	no	video	no
Speaker Diarization Toolkit	C/C++	yes	audio	
iBUG TAUD	Matlab	no	video	no
iBUG Gesture Detector	Matlab	no	video	no
Active appearance models	Matlab	yes	video	no
AAM-FPT Facial Point Tracker	C/C++	no	video	no
BoRMaN	Matlab	no	video	no
iBUG-GENOSHA	Matlab	no	video	no
Gesture detector	Matlab	no	video	no
LAUD	Matlab	no	video	no
ViBe - background substraction tool	C/C++	partly	video	yes
Gaze and Head Pose Estimation	C/C++	yes	video	yes
SEMAINE Research Platform	C/C++	partly	multimodal	yes
Salient Point Detector	Matlab	no	video	no
Gabor Facial Point Detector	Matlab	no	video	no

Table 4.2: Tools available on *SSPNet* (retrieved April 2015). Almost all of them are tuned for a single modality and are not meant for online processing.

not meant for interaction. Or, removing those parts of a recording which are impaired by noise or partly lack sensor information might suit high recognition accuracies, but miss issues that are relevant in online applications, where sensor failure and unfavourable conditions cannot be avoided per se. Finally, other issues, such as latencies or synchronisation issues, are likely to be underrated. Mainly drawing on offline studies, is another reason why – despite the great effort accomplished in the last years – real-life applications are still a curiosity. This is also reflected in the list of hosted tools on the *SSPNet Portal*[10] listed in Table 4.2. We nfote that with a few exceptions most of the code is closed source. However, even more importantly only two in the list are actually online systems, while the others are offline tools mainly written in Matlab[11].

[10] `http://sspnet.eu`
[11] `http://www.mathworks.de/`

4.3.7 Lessons

For different reasons shifting from offline to online processing is not a trivial task. One particular challenge arises from the fact that there is no simple way to skip "inconvenient" data. In fact, a signal has to be processed as it comes, which requires special strategies to handle the case when information becomes corrupted or meaningless. To account for this **methods for handling missing data** and possibilities to **dynamically decide when to trust data** are required. In particular, a **garbage class is needed to intercept information that is not relevant** to the current task.

The continuous evolvement of social behaviour not only calls for continuous expert coding (see Section 4.1.5), but also demands from a recognition system to output predictions in a steady and dynamic manner. Hence, instead of decomposing an input signal into discrete and isolated parts and assigning each segment to one out of a set of fixed categories, a **continuous classification** is required which updates the current state at a regular time basis on a dynamic scale. This calls for new classification schemes embedded in a flexible architecture which not only gives support for conventional statistic-based classification, but also dynamic classifiers, such as Hidden Markov Models (HMMs) or Artificial Neural Networks (ANNs). Also, special attention has to be paid to the segmentation problem since detected **boundaries have to suit recognition but also allow for a fluent interaction**.

To evaluate the quality of an online recogniser conventional measurements, which draw on a sample-wise comparison, are no longer feasible. Instead a **metric is required that takes into account the temporal relation between ground truth and prediction**.

The most promising way to assure that spent efforts will translate into a real-time capable application is by **developing towards an online system from the very beginning**. That is, instead of designing and testing a system offline and converting it afterwards, an environment is needed that allows developers to test implemented algorithms in an online setting straightaway. However, this should not demand a great deal of additional work, but mainly prevent from designing algorithms that are not real-time capable per se, e.g. by ruling out access to future data. Any generic task, such as threading and buffering the raw and processed data as well as storage and visualisation, should be handled automatically without further ado. To further cut down development time the distribution of re-usable software should be fostered. To this end, facilities should be provided that encourage developers to **reuse and modularly expand the pool of available solutions**, whereas end users should be offered an **easy-to-use interface for setting up systems** without demanding certain programming skills and third-class development tools.

Chapter 5

The Social Signal Interpretation Framework

In this chapter we introduce SOCIAL SIGNAL INTERPRETATION (SSI) [331, 333, 336], a framework for recording, analysing and fusing social signals in real-time. It has been designed to provide a general architecture to tackle the challenges we have discussed in the last chapter. To account for future developments and trends universal solutions are preferred over specific ones. For instance, signals are handled by the general concept of a *stream* independent of the actual content.

In the previous chapter three core challenges have been identified and it was argued that solving those will help fostering the development of real-life applications capable of reliably extract social and affective signals. In short these are:

"collection of large and rich multimodal corpora"

As we will see, SSI allows synchronised reading from a variety of different sensing devices, such as audiovisual sensors, eye trackers, physiological feedback systems, motion capture suits, etc.. To account for future technology SSI provides a flexible interface to ingrate novel sensory.

"investigation of advanced fusion techniques"

Human interaction is characterised by complementary, redundant and contradictory cues, which calls for smart ways of fusing information from multiple modes. SSI supports fusion of multimodal information at different stages, e.g. following a conventional segmentation-based approach. However, SSI's flexible architecture also suites asynchronous and event-based fusion.

"simplifying the development of online systems"

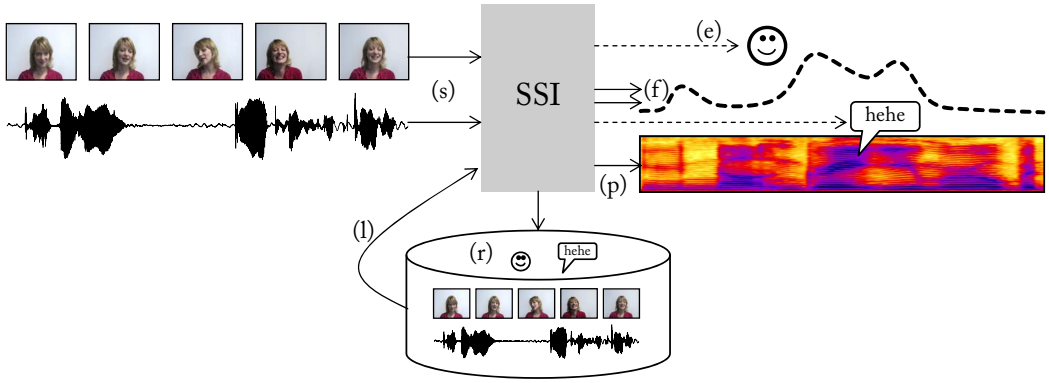


Figure 5.1: SSI allows the synchronised reading from multiple sensor devices (s). Signal data can be processed continuously (p) or in form of high-level events (e). Information from multiple sources can be fused along the way (f). Raw and processed data (in form of streams or at event-level) can be recorded (r) and stored for later analysis and model learning (l).

Shifting from offline to online processing introduces a bunch of new challenges. SSI provides a framework to apply on-the-fly signal processing and pattern recognition to extract higher-level information in real-time. Important concepts such as introduction of a garbage class or replacing static categories with continuous dimensions are integral parts of the the system architecture. Experts and novice users are provided with appropriate tools to put together pipelines from single, reusable processing units.

The following chapter deals with the basic concepts of the SSI framework, in particular, the processing, buffering, and synchronisation of signal streams and the various levels at which information can be fused. Theory on real-time signal processing and machine learning strategies will be given in the appropriate place.

5.1 Core Design

Before going into details, this section summarises the main thoughts and goals that have driven the design of the framework. The core functions of SSI are visualised in Figure 5.1

5.2 Generic Data Handling

Sensors and the physical quantities they measure build the foundation of SSI. Through repeated metering a sensor produces a series of measurements and an elegant way is needed to represent them. Since value type as well as update rate vary depending on the observed quantity, we need a generic solution that is not biased towards a certain type of signals.

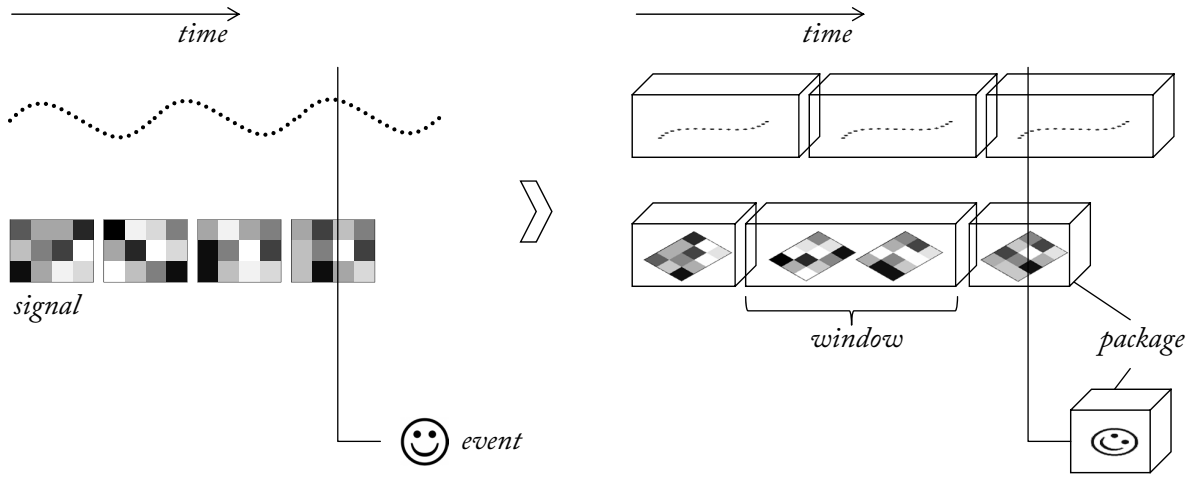


Figure 5.2: A generic data structure masks signals and events. To account for individual window lengths packages are of variable length.

For instance, let us consider the properties of a video signal versus that of a sound wave. The images in a video stream are represented as an assembly of several thousand values expressing the colour intensity in a two-dimensional grid. The rate at which images are updated is around 30 times per second. A sound wave, on the other hand, consists of single values quantifying the amplitudes, but is updated several thousand times per second. A tool meant to process video images will therefore differ very much from a tool designed to work with sound waves. To process a video stream we could grab a frame, process it, grab the next frame, process it, and so on. In audio processing such a sequential approach is not applicable because signals values have to be buffered first. And since update rates between values are much shorter and buffering cannot be suspended during processing, the two tasks have to be executed in parallel.

To deal with such differences, raw and processed signals should be represented using a generic data structure that allows handling them independently of origin and content. This can be achieved by splitting signals into smaller parts (*windows*) and wrap them in uniform packages of one or more values. To account for individual processing timing, packages can be of variable length. Treating signals as sequences of "anonymous" packets has the advantage that any form of buffering and transportation can be implemented independently of the signal source. The same kind of concept can be applied to handle gestures, key words and other higher level information which is not of a continuous nature. A generic wrapper for discrete events makes it possible to implement a central system to collect and distribute events. This allows for an environment that works with virtually any kind of continuous and discrete data produced by sensors and intermediate processing units (see Figure 5.2). A useful basis for a framework intended to process multimodal sensor data.

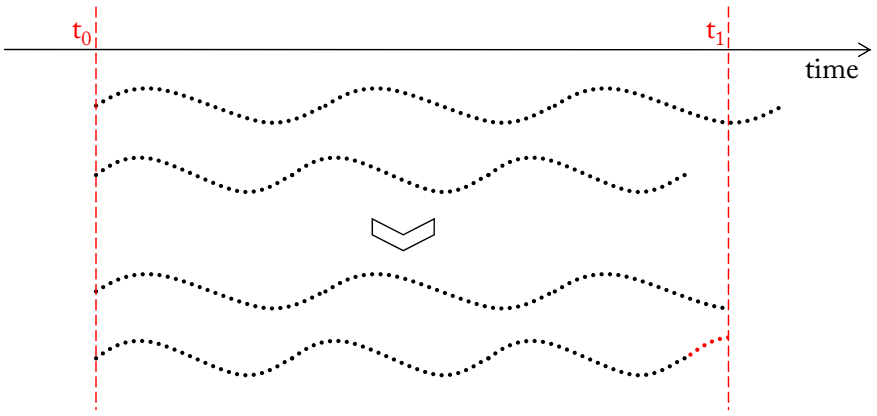


Figure 5.3: To keep signals in-sync in regular intervals additional/missing values are removed/added.

5.2.1 Synchronization

When working with multimodal data we want to decide which parts of two signals temporally correspond with each other. For instance, we would like to know which sound occurred together with a certain video frame. Media players need to solve this problem – known as *audio/video sync* – to prevent that sound and image drift apart. Typically it is solved by either interleaving video and audio data or by explicit relative time-stamping. In case of *interleaving* synchronisation is ensured by alternately storing small parts of the signals. For instance, we could bundle an image with the sound chunk captured before the next frame is retrieved. Making each image a synchronisation step effectively prevents that the two channels will drift apart. Alternatively, signals can be *time-stamped* according to a common clock, so that a later mapping is enabled.

But is interleaving or time-stamping a suitable technique to synchronise signals in a general multimodal framework? Interleaving means to facilitate a joint processing of data, which is not a practical solution for a framework in which signals have to be handled independently, too. Apart from that, signals have to be stored in one common file, which makes it difficult to parse them afterwards. Since the creation of multimodal databases is a core task of SSI, stored signals should be easy to access from other applications, preferably in formats that are widely supported. Although there are some standardised container formats such as *Audio Video Interleaved* (AVI) (see Section 5.5), those are dedicated to specific signal types of signals and not suited as a general storage format. Hence, it is preferable to store signals in separate files. Now, what about time-stamping? Although possible it is not very convenient either. To jump to a certain point in time we first find the nearest common time-stamp and then interpolate to the requested position. The main drawback of this approach is that synchronisation is warranted only as long as time-stamps are available.

So is there are a way to process signals separatively without storing time-stamps? It is: but we have to make sure signal values are produced at precise intervals. As explained in more detail in

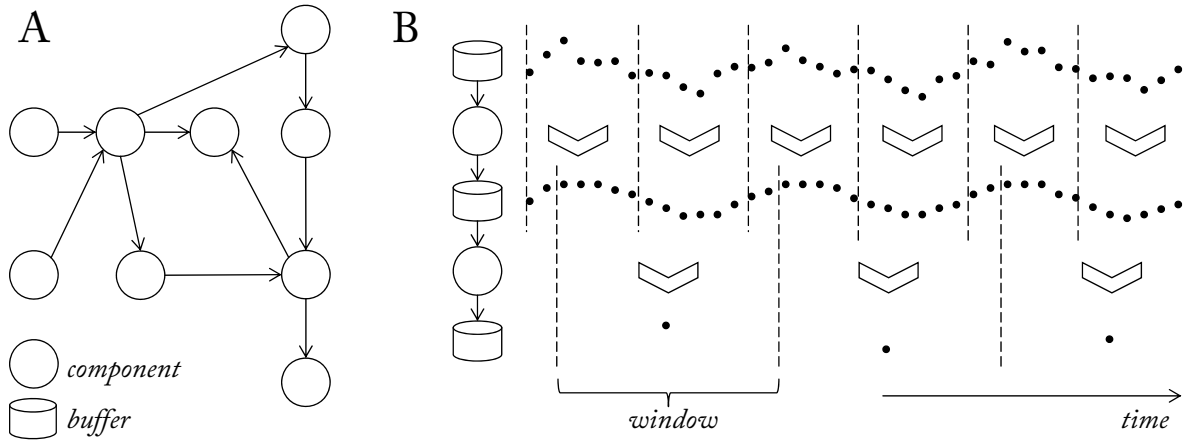


Figure 5.4: Complex recognitions task are distributed across individual components organised in a directed acyclic graph (A), called pipeline. The output of a component is buffered so that other components can access it at individual window sizes (B).

Section 5.3.2 digital signals are made of discrete values that are generated according to a *sample rate*. By means of the sample rate it is possible to calculate the expected number of values in a certain span. In practice we will find that this number is not always matched due to inaccuracies of the sensors. However, since we can calculate the relative drift of the signal, synchronisation can be retained by removing redundant values or adding missing ones (see Figure 5.3). Although this alters the signal, changes are barely perceptibly if adaptation is applied in sufficiently short intervals. The exact approach is elucidated in Section 5.4.3. It ensures that signals in SSI are kept in sync at any stage of processing, which provides the basis for any kind of information fusion, let it be on an early or late stage.

5.2.2 Modular Design

Usually raw sensor data does not give away the information we are interested in without further ado. For instance, if we want to analyse the affective state of the user in a video we locate the position of the face, extract features that describe the face in terms of few meaningful parameters, and finally, apply a model that maps the features to a concrete affective state (see Section 3.2). In practice, more steps might be required. What matters is that in this way a complex procedure is split into a series of clearly defined steps, which in the following we will refer to as a system's *components*.

Building up systems from individual components according to a *modular design* highly improves maintainability. This is because components introduce a level of abstraction in which function is separated from implementation. If, for instance, an improved version of an algorithm becomes available, it only concerns the back-end of the component, while the visible interface remains

the same. Hence, components can be exchanged without the necessity to adapt other parts of the system. Likewise, a component can be reused in a different context.

Of course, decomposing a system into independent components requires additional infrastructure. An order has to be defined to describe the interaction between components. A sequence of at least two connected components is called a *pipeline*. In the simplest form it is realised as a single chain in which a component receives data, processes it and hands it over to the next component. However, to process a single data stream in different ways, it should be possible to branch out into multiple forks. Likewise, since we want to be able to fuse data, it should be possible to join forks, too. An appropriate mathematical construct to describe such pipelines is a directed acyclic graph, i.e. a directed graph with no directed cycles, in which data is propagated along its edges.

Of course, we also need to define how data is passed from one component to another. Data can be either passed directly from one component to another, or it can be cached in an interposed buffer. The latter has the advantage that data can be accessed independently of the producer, which makes it usable with different window sizes as depicted in Figure 5.4. To give an example, facial expression analysis may start from a frame-to-frame base, where each frame is analysed independently. At later stages, processing may rely on bundles of frames to capture temporal dynamics.

5.2.3 General Methodology

To reach a large community, a methodology is needed that addresses developers interested in extending the framework with new functions as well as end-users whose primary objective is to create processing pipelines from what is there. Again, a modular design pays off as it allows components to be classified into few general classes, such as “this-is-a-sensing-component” and “this-is-a-transforming-component”. By masking individual differences behind a few basic entities it becomes possible to translate a pipeline into a another representation (and the other way round). This can be exploited to create an interface which allows end-users to create and edit pipelines outside of an expensive development system and without the knowledge of a complex computer language.

Developers, on the other hand, should be encouraged to enrich the pool of available functions. Therefore an API should be provided which defines basic data types and interfaces as well as tools to test components from an early state of development. In particular, the simulation of sensor input from pre-recorded files becomes an important feature, as it allows for a quick prototyping without setting up a complete recording setup, yet providing realistic conditions, e.g. by ruling out access to future data. Since all data communication is shifted to the framework,

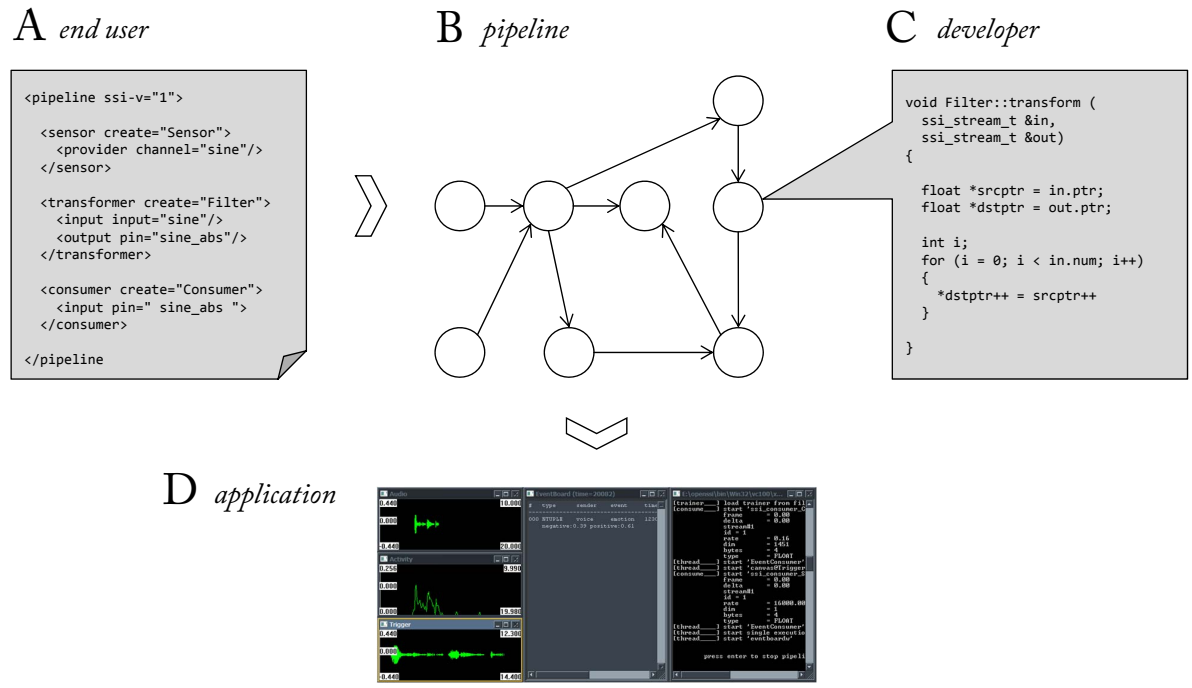


Figure 5.5: General methodology: a simple markup language allows end-users to connect components to a pipeline. An interpreter translates the structure, connects the sensor devices and starts the real-time processing. Developers are encouraged to implement new components and add them to the pool of available nodes.

additional efforts are minimised. The general methodology is depicted in Figure 5.5 and will be addressed in detail in Section 5.6.

5.2.4 Comparison with OpenSMILE

Before we go into details on the implementation of SSI it is worthwhile to draw a comparison with the open-source feature extraction toolkit OPENSMILE [107, 109]. Together with OPENCV[1], which is a library of programming functions mainly aimed at real-time computer vision, OPENSMILE is one of the rare solutions available to the research community for building live demonstrator systems. In fact, over the past years, OPENSMILE has established as a standard tool for the extraction of paralinguistic features, in particular it is regularly used to compute the baseline for the annual INTERSPEECH challenge in the area of Computation Paralinguistics [290, 292, 294–297].

In many respects, OPENSMILE and SSI share a similar philosophy and several core features can be found in both implementation.

1. Both architectures have been designed with a strong focus towards real-time and incre-

[1] http://opencv.org/

mental processing.

2. C++ has been chosen as an efficient low-level programming language.
3. The concept of a circulate, or ring buffer has been adopted to handle continuous data streams without constantly allocating and deleting memory (Section 5.3.4).
4. Data processing is distributed over several independent components which are separately maintained and executed in parallel (Section 5.4.2).
5. Processed data can be shared among multiple consumer to avoid duplicated computations (Section 5.4).
6. A plugin interface is offered to implement new functions and extend the pool of available components (Section 5.6.7).
7. Components are configured through a list of options and a script language is supported to define the processing chain (Section 5.6.7).

However, there are also some fundamental differences. Although it is principally possible to process other modalities in OPENSMILE and in fact, since version 2.0 some basic video processing based on OPENCV has been added, its primary focus is the extraction of audio features. In comparison, SSI follows a broader objective.

1. Apart from audiovisual sensory, a wide range of devices are supported, including physiological sensors, (stationary and mobile) eye tracking hardware, motion capture suites, data gloves, depth sensors, pressure mats, ...
2. Realisation of complex multimodal recording setups, possibly distributed over several machines in a network, and mechanisms to keep captured data in sync without time-stamping (Section 5.4.3).
3. On top of a frame-by-frame based processing adding support for an asynchronous event-based handling of higher level information (Section 5.4.4).
4. Full integration of the complete machine learning pipeline including the possibility to record, train and test classification algorithms in an online environment (Section 5.4.5 and Chapter 6).
5. Providing a rich test-bed for developing novel fusion algorithms (Section 5.4.6 and Chapter 7).

In fact, a wrapper is available in SSI to extract audio features with OPENSMILE and fuse them with information from other channels.

5.3 Signals

Signals are the primary source of information in SSI. Basically, a signal conveys information about some physical quantity over time. By re-measuring the current state in regular intervals and chronically stringing these measurements we obtain a signal. In the following we will refer to a single measurement as a *sample* and denote a string of samples as a *stream*. By stepwise modifying the signals, we try to carve out information about a user's social behaviour. This is what social signal processing is about and it is the job of SSI to make it happen.

5.3.1 Sensing

Above all, we need sensors. A *sensor* (also called *detector*) is a device that measures a physical quantity and converts it into a signal. Observing the signal allows us to draw conclusions about the state of the observed property. For instance, from the state of the mercury inside a calibrated glass tube we can read off the temperature of its surrounding. Ideally, a sensor should be sensitive towards the measured property without influencing it or being sensitive to other properties. The output of the sensor should be a simple function of the measured property, e.g. linear or logarithmic. The sensitivity of a sensor is then the ratio between output signal and measured property, and the smallest change that can be detected in the measured quantity defines the resolution. Although we are often not aware of it, sensors play an important role in our everyday life and we are constantly making use of them. Modern cars, for instance, include several hundred sensors from oxygen sensors, over crank sensors, mass air flow metres and coolant temperature sensors.

Nowadays, also a wide range of sensors are available that deliver digital output and can be plugged to a computer. Today the common way to connect a device is via *Universal Serial Bus* (USB). USB was designed in the mid-1990s to standardise the connection of computer peripherals, such as keyboards, printers, and disk drives, but also for digital cameras, microphones, eye tracker, and other sensors. It is used for both, data transmission and to supply electric power, and effectively replaced a variety of earlier interfaces, such as serial and parallel ports. Since USB requires a hardwired connection with the computer and allows only limited mobility, it is not always a suitable choice. Wireless transmission protocols offer an alternative. Among wireless technology, *Bluetooth* is the by far most common standard. It allows exchanging data over short distances using short-wavelength Ultra high frequency (UHF) radio waves.

But first, analog signals need to be converted into a digital representation. A process we denote as *sampling*.

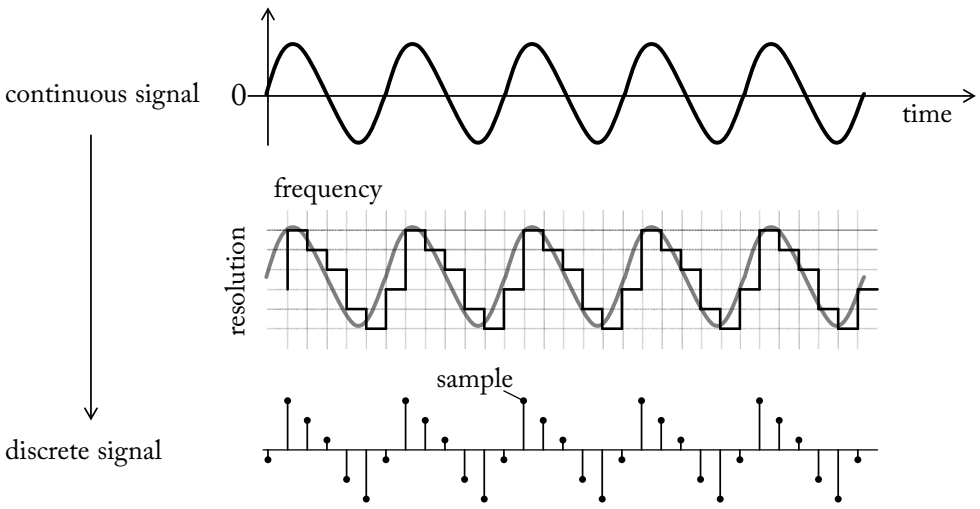


Figure 5.6: *Sampling* is the process of converting a continuous signal (top) to a time-series of discrete values (bottom). Two properties characterise the process: the frequency at which the signal is sampled (*sampling rate*) and the number of bits reserved to encode the samples (*sample resolution*)

5.3.2 Sampling

Basically, an analog signal is a continuous representation of some quantity varying in time. Since an analog signal has a theoretically infinite resolution it would require infinite space to store it in digital form. Hence, it is necessary to reduce the signal at discrete points of time to discrete quantity levels. To do so an analog signal is observed at fixed time intervals and the current value is quantised to the nearest discrete value of the target resolution. The process is visualised in Figure 5.6. The graph shows a continuous signal (top) that is reduced to a series of discrete *samples* (bottom). In a digital world any signal is represented by a finite time-series of discrete samples. Following from the sampling procedure a digital signal is characterised by the frequency at which it is sampled, the so called *sampling rate*, and the number of bits reserved to encode the samples, the so called *sample resolution*.

Both, sampling rate and sample resolution limit the information kept during conversion. For instance, given a resolution of 8 *bit* we can encode an analog input to one in 256 different levels. If the resolution is good enough to capture sufficient information about the signal, or if we have to enlarge the value range by allocating more bits, depends on the measured quantity. We can check it by determining the quantisation error, which is the difference between an analog value and its quantised value. The useful resolution is limited by the maximum possible *signal-to-noise ratio* (S/R) that can be achieved for a digitised signal. S/R is a measurement for the level of a desired signal to the level of background noise. If the converter is able to represent signal levels below the background noise additional bits will no longer contribute useful information.

Likewise we can also estimate the sample rate. According to the *Nyquist-Shannon sampling theorem* a perfect reconstruction of the analog signal is (at least in theory) possible if the sampling

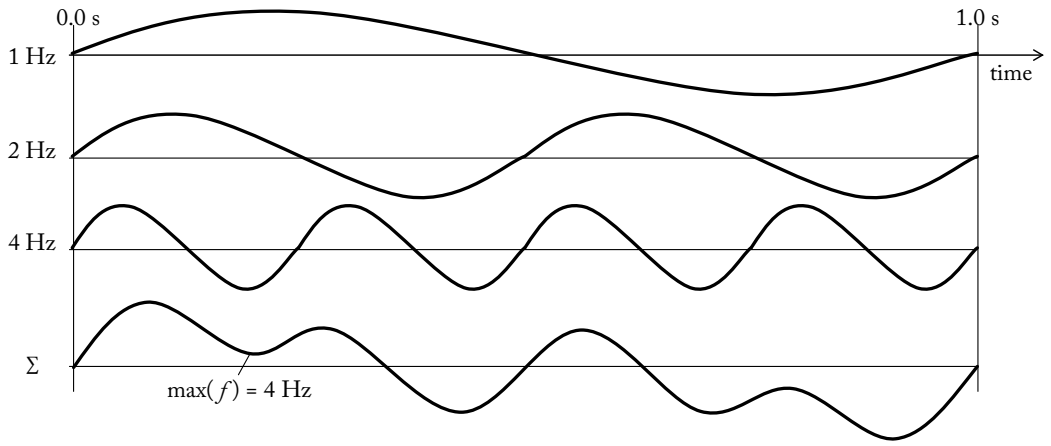


Figure 5.7: A periodic signal is a signal that repeats a certain pattern over and over again. The completion of a full pattern is called *cycle*. By counting the cycles per seconds we can measure the *frequency* of the signal. The graph shows sine waves with frequencies of 1, 2, and 4 Hz ($= \frac{1}{second}$). The sum of the three sine waves (bottom graph) is again a periodic signal and has a *highest frequency* equal to the largest single frequency component, that is 4 Hz.

rate is more than twice as large as the highest frequency of the original signal, the so called *Nyquist frequency*. To understand this relation, we first have to know what is meant by *highest frequency*. Let us start by defining a periodic signal. A periodic signal is a signal that completes a pattern after a certain amount of time and repeats that pattern over and over again in the same time frame. The length of the time frame in seconds is called *period* and the completion of a full pattern is called *cycle*. If the signal is a smooth repetitive oscillation, e.g. a *sine wave*, we can determine its *frequency* by counting the number of cycles per seconds. It is measured in units of *Hertz* ($Hz = \frac{1}{second}$). Figure 5.7 shows sine waves with frequencies of 1, 2 and 4 Hz. If we sum up the samples of the sine waves along the time axis we get another periodic signal (bottom graph). The *highest frequency* of the combined signal is equal to the largest single frequency component, that is 4 Hz. In fact, any periodic function can be described as sum of a (possibly infinite) set of sine waves (a so called Fourier series). According to the *Nyquist-Shannon sampling theorem* we must therefore sample the summed signal at a sample rate greater 8 Hz.

To illustrate the relation between the Nyquist frequency and the sample rate we can think of a periodic signal swinging around the zero axis. If the signal completes on cycle per second, i.e. has a (maximum) frequency of 1 Hz, we will observe in every second a peak when signal values are above zero and a valley when signal values are below zero (see top graph in Figure 5.7). If the signal is sampled at a sampling rate of 1 Hz, i.e. we keep only one value per cycle, we pick either always a positive or always negative value. Obviously, we will not be able to correctly reconstruct peaks and valleys. If we increase the sample rate to 2 Hz, i.e. we sample twice per second, we have a good chance to get in each cycle a positive and a negative value. However, it may happen that we pick twice in the moment where the signal crosses the zero axis. In this case the sampled signal looks like a zero signal. Only by choosing a sample rate greater 2 Hz we

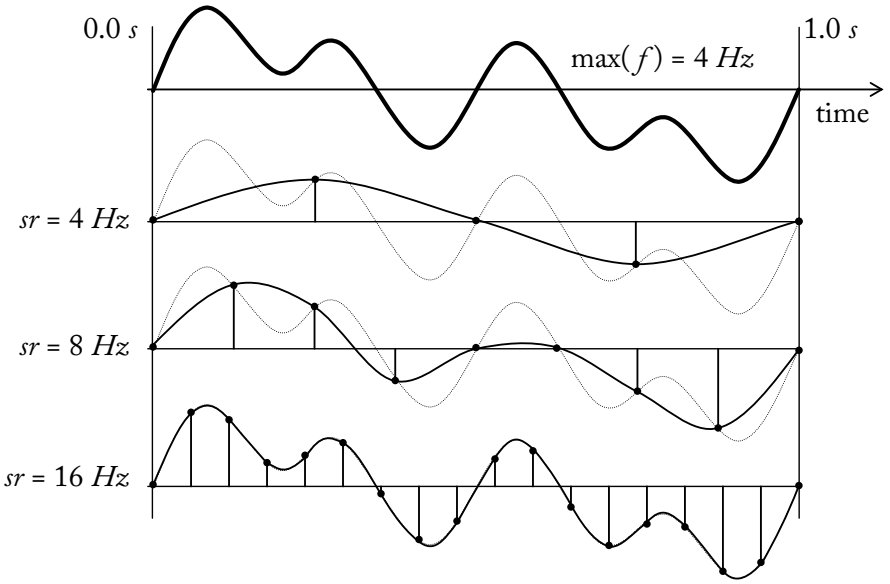


Figure 5.8: According to the *Nyquist-Shannon sampling theorem* a signal can be reconstructed if the sampling rate (*sr*) is more than twice as large as the highest frequency (*Nyquist frequency*) of the original signal. The example shows an periodic signal with a highest frequency of 4 *Hz* sampled at different rates. Only for the last case, where the sample rate is above the Nyquist frequency (8 *Hz*), the original signal can be correctly reconstructed.

can ensure to pick at least one value from a peak and one value from a valley. Figure 5.8 shows the summed signal from the previous example sampled at 4 *Hz*, 8 *Hz* and 16 *Hz*. Only in the last case, where the sample rate is above the Nyquist frequency (8 *Hz*), the original signal can be reconstructed.

We have seen that sampling rate and sampling resolution need to be carefully chosen to neither waste space, nor to lose information. From now on we always implicitly refer to digital signals when talking about signal streams.

5.3.3 Streaming

In SSI, a *stream* is defined as a snapshot of a signal in memory or on disk, made of a finite number of samples. The samples in a stream are all of the same kind. In the simplest case a sample consists of a single number, which we refer to as a *sample value*. It can also be an array of numbers and in this case the size of the array defines the *dimension* of the sample. Instead of numbers the array may also contain more complex data types, e.g. a grouped list of variables. The number of samples in the stream, the sample dimension and the size of a single value in *bytes* are stored as meta information, together with the sample rate of the signal in *Hz* and a time-stamp, which is the time difference between the beginning of the signal and the first sample in the stream. A stream is represented by a reference to a single memory block which holds the sample values stored in interleaved and chronological order. The total size of the data block is

derived as the product of the number of samples $\times$ the sample dimension $\times$ the size of a single value.

Let us consider some examples: on the top of Figure 5.9 a stream storing the position of a mouse cursor for 1 *s* is shown. The sample rate is 5 *Hz*, which means the cursor position is scanned 5 times within one second. Since the position of the mouse cursor is reported in x and y coordinates the stream has two dimensions. To store each coordinate 2 *bytes* (*short integer*) are reserved. In total the stream data measures 20 *bytes* (5 samples $*$ 2 dimensions $*$ 2 *bytes*). Although a single time-stamp is assigned for the whole stream we can easily give time stamps for each sample by adding the product of sample index and the reciprocal of the sample rate ($\frac{1}{5Hz} = 0.2$ *s*). For example, the last sample in the stream (index 4) is assigned a time-stamp of 3.8 *s* ($3.0 + 4 * 0.2$ *s*). This way of deriving time stamps make it redundant to store time stamps for all except the first sample.

Now, let us think of an audio signal (centre of Figure 5.9). In due consideration of the Nyquist-Shannon sampling theorem and since the human hearing covers roughly 20 to 20,000 *Hz*, audio is typically sampled at 44,100 *Hz*[2]. The sample resolution is usually 16 *bits* (2 *bytes*), which yields a theoretical maximum S/R of 96 *dB*[3]. This is sufficient for typical home loudspeakers with sensitivities of about 85 to 95 *dB*. The audio stream in Figure 5.9 stores a mono recording, hence stream dimension is set to 1. If it was stereo dimension would be 2. The sample values are integers within a range of -32,768 to 32,767, which exploits the full resolution of $2^{16} = 65,536$ digits.

Finally, on bottom of Figure 5.9 a gray scale video stream is given. For demonstration purposes the video images have a resolution of 4×3 pixel, i.e. an image consists of 12 gray scale values. It may seem surprising that the stream dimension is still 1 not 12. This becomes clear if we consider a stereo camera which delivers images in pairs. If the dimension would be the sum of pixels from both images we could no longer decide whether it is two images or a single image with twice as many pixels. Hence, to avoid ambiguities we treat each image as a single sample value. Given that the gray scale information of a pixel is encoded with 1 *byte* the size of a sample value is 12 *bytes*. Thus, the total size of the video stream is 300 *bytes* (1 *s* $*$ 25 *Hz* $*$ 12 *bytes*). Since a standard webcam delivers images in RGB or YUV (3 *bytes* per pixel) at a resolution of 320×240 pixels up to 1600×1200 pixels the space to store of a single sample value can actually take up several *MB*, e.g. $1600 * 1200 * 3$ *bytes* $= 5,760,000$ *bytes* $= 5.76$ *MB*.

As the examples show the implementation of streams in SSI follows a very generic concept. The same data structure is used to handle literally any kind of signal, regardless of whether a

[2]A sample rate of 44,1 *kHz* is common for conventional audio CDs. For the transmission of speech. (e.g. in telephony) the sample rate can be reduced to the frequency range of the human voice, which is from approximately 300 *Hz* to 3400 *Hz*. Hence, a sampling rate of 8 *kHz* is sufficient.

[3]For each 1-*bit* increase in bit depth, the S/N increases by approximately 6 *dB*.

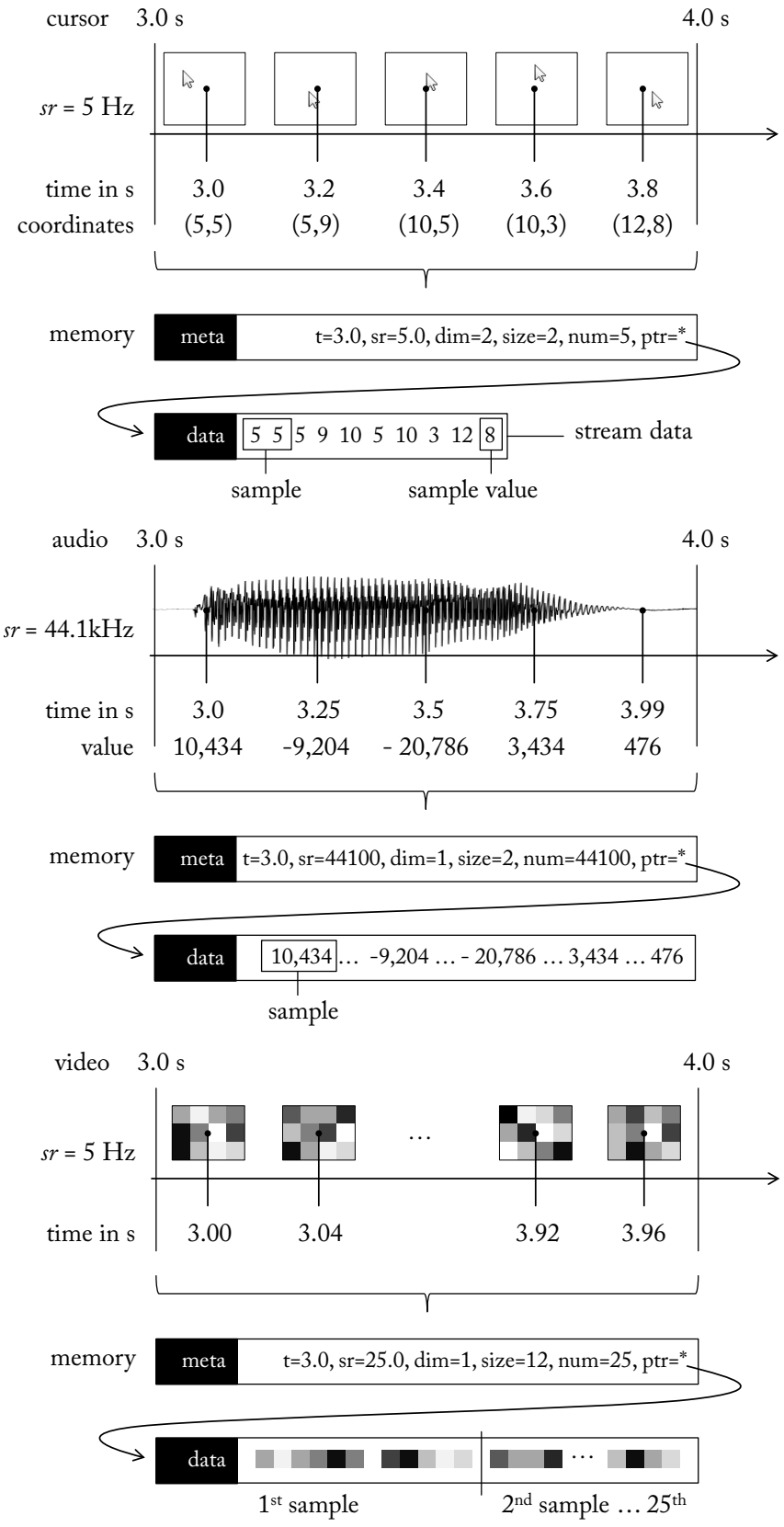


Figure 5.9: The examples show how SSI's stream structure is applied to different quantities: a cursor signal, an audio chunk, and a video stream.

cursor signal, an audio signal, or a video signal has to be transmitted. This allows for a coherent handling and transmission of streams independent of their content.

5.3.4 Buffering

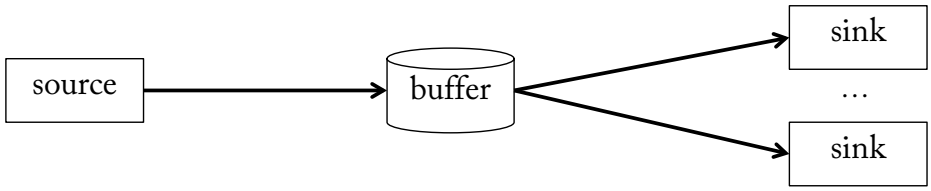


Figure 5.10: Signals are exchanged through buffers which allow *sinks* to access the output of a *source*.

Signal flow describes the path a signal takes from source to output. As described, streams offer a convenient way to implement this flow as they allow handling signals in small portions. If we define a *source* as an entity which outputs a signal and a *sink* as an entity which receives it, a natural solution would be to simply pass on the data from source to sinks. However, in this case the intervals at which a sink is served would be steered by the source. This is not feasible, since a sink may prefer a different timing. For example, a source may have a new sample ready every 10 ms, but a sink only asks for input every second. Hence data flow should be delayed until 100 samples have been accumulated. This can be achieved by temporarily storing signal samples in a *buffer*.

A buffer knows two operations: either samples are written to it, or samples are read from it. It is tied to a particular source and signal, i.e. it stores samples of a certain type, and can connect one ore more sinks (see Figure 5.10). Since memory has a finite size, a buffer can only hold a limited number of these samples. When its maximum capacity is reached there are two possibilities to choose from: either enlarge the buffer, which only pushes the problem one stage back, or to sacrifice some samples to make room for new ones. The latter is exactly the function of a so called *circular buffer* (also *ring buffer*).

A circular buffer is a data structure that uses a single, fixed-size buffer as if it were connected end-to-end. A circular buffer has the advantage that elements need not be shuffled around when elements are added. It starts empty pointing to the first element (*head*). When new elements are appended the pointer is moved accordingly. Once the end is reached the pointer is again moved to the first position and the buffer begins to overwrite old samples (see Figure 5.11). The simple logic of a circulate buffer suites a highly efficient implementation, which is important given the high frequency of read and write operations a buffer possibly has to handle.

When a sink reads from a buffer, it sends a request to receive all samples in a certain time interval. If the data is available a stream including a copy of the samples is returned, i.e. during read op-

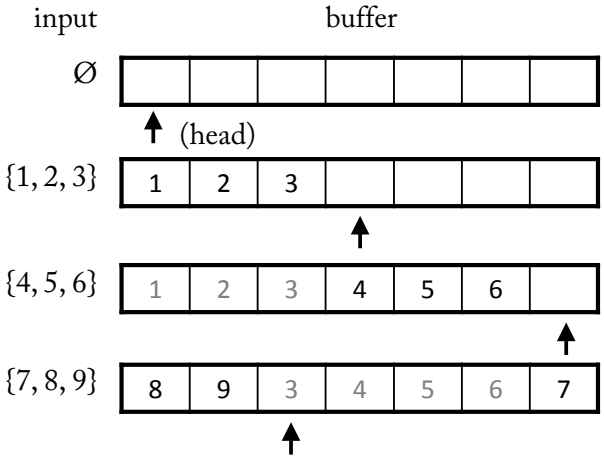


Figure 5.11: A circular buffer starts empty pointing to the first element (head). When new elements are appended the pointer is moved accordingly. Once the end is reached the pointer is again moved to the first position and old elements are overwritten.

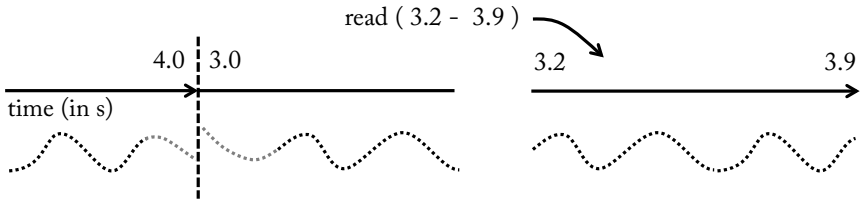


Figure 5.12: During read operations samples are copied, so that the content of the buffer remains unchanged.

erations the content of a buffer is not changed (see Figure 5.12). Hence, multiple read operations are supported in parallel. During a write operation, on the other hand, a stream with new samples is received by the buffer, which will possibly replace previous samples. Consequently, writing samples to a buffer alters its content and therefore should be handled as an atomic, i.e. exclusive, operation. This is achieved by locking the buffer as long as write operation is in progress, which guarantees that no read operations occur in the meanwhile. Figure 5.13 shows the content of a buffer before and after a write operation. If we compare the unfolded streams we see that new samples were appended to the front of the stream at cost of samples at the ending.

5.4 Pipelines

On October 1, 1908, the Ford Motor Company released "Model T", which is regarded as the first affordable automobile and became the world's most influential car of the 20th century. Although Model T was not the first automobile it was the first affordable car that conquered the mass market. The manufacturing process that made this possible is called an *assembly line*. In an assembly line interchangeable parts are added as the semi-finished assembly moves from work

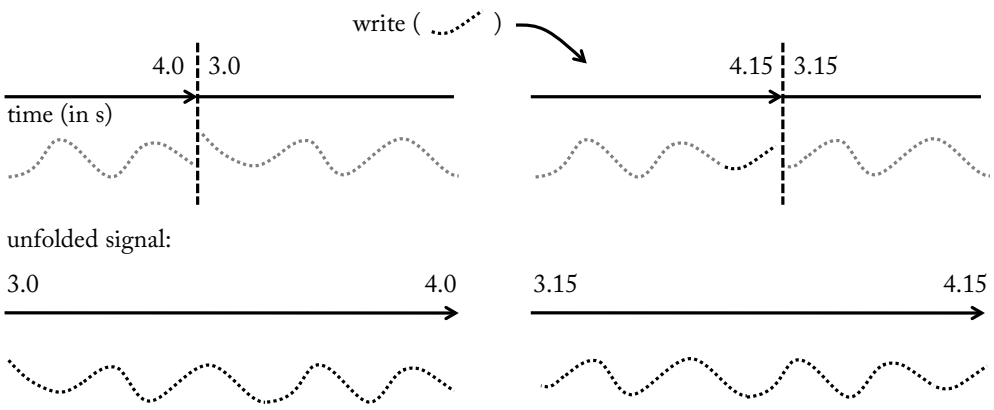


Figure 5.13: During a write operation samples are appended, which alters the content of the buffer.

station to work station where the parts are added in sequence until the final assembly is produced. The huge innovation of this method was that car assembly could be split between several stations, all working simultaneously. Hence, by having x stations, it was possible to operate on a total of x different cars at the same time, each one at a different stage of its assembly.

The same kind of technique is adopted in SSI to achieve an efficient processing of the signals. Work stations are replaced by *components* which receive and/or output one or more streams, possibly altering the content. By putting multiple components in series, a processing chain is created to transform raw input into something more useful. And like in the case of Henry Ford's assembly lines, the components can work simultaneously. We call such a chain *pipeline*. Since a pipeline can branch out into multiple forks, but also join forks, it represents a *directed acyclic graph*.

5.4.1 Signal Flow

A pipeline is a chain of processing components, arranged so that the output of each component feeds into successive components connected by a buffer (see previous Section). Although such an intermediate step introduces a certain amount of overhead caused by copy operations during read and write operations, it bears several advantages. First of all, the output of a component can be processed in parallel by several independent components and, as already pointed out earlier, read and write operations can be performed asynchronously. To actually make this possible, SSI starts each component as a thread. This is an important feature, since components in front of the the pipeline often work on high update rates of a few milliseconds, whereas components towards the end of a pipeline operate at a scale of seconds. The length at which a signal is processed is also called *window* length. Generally, we can say that the window length in the pipeline grows with position. Apart from offering more flexibility this also has practical advantages, since components further back in the pipeline cannot cause a delay in the front of the pipeline,

which may lead to data loss[4]. Finally, running components in different threads helps to make the most of multi-core systems.

An efficient handling of the signal flow between the components of a pipeline is one of the challenges to a real-time signal processing framework. The problems to be dealt with are similar to those in a *consumer-producer problem* (also known as *bounded-buffer problem*). The problem describes two processes, the *producer* and the *consumer*, who share a common, fixed-size buffer. The producer generates data and puts it into the buffer. The consumer consumes data by removing it from the buffer. Hence, the producer must not add data into the buffer if it is full and the consumer must not try to remove data if the buffer is empty. A common solution would be to put the producer to sleep if the buffer is full and wake it up next time the consumer begins to remove data again. Likewise the consumer is put to sleep if it finds the buffer to be empty and awaked when the producer starts to deliver new data. An implementation should avoid situations where both processes are waiting to be awakened to not cause a deadlock. And in case of multiple consumers what is known as starvation, which occurs if a process is perpetually denied data access.

Transferred to the problem at hand there are two differences. First, we can always write to a circular buffers since old values are overwritten when the buffer is full. And second, components do not remove data but get a copy. So except for the beginning a buffer is never empty. However, if a component requests data that already have been overwritten, we still encounter the situation where the requested data cannot be delivered. Either because a component requests samples for a time interval that is not yet fully covered by the buffer. In this case we should put the calling component in a waiting state until the requested data pops up. Or because the requested samples are no longer in the buffer. In this case the operation will never succeed so we should cancel it. As already pointed out earlier write requests should be handled as atomic operations. To prevent starvation, waiting components are put in a queue and awakened in order of arrival.

At run-time, components are operated at a *best effort delivery*, which means data is delivered as soon as it becomes available. When a component receives the requested stream it applies the processing and returns the result. Afterwards it is immediately put on hold for the next data chunk. Ideally, the first component in a pipeline functions as a sort of bottle neck and following components finish in average before the next chunk of samples becomes available. This guaranties that the pipeline will run in real-time. However, since stream data is buffered for a certain amount of time, components are left some margin to finish their task. The window length at which a component operates depends on the algorithm, but also the kind of signal. An audio stream, for instance, which has a high sample rate, is usually processed in chunks of several hundred or even thousand samples. Video streams, on the other hand, are often handled on a frame-by-frame base. While the number of samples in a stream may vary with each call,

[4]Of course, this assumes a proper buffering of intermediate results generated by components from the front until they are ready to be processed by the slower components in at the end.

sample rate and sample dimension must not change. Due to the persistence of streams most resources can be allocated once in the beginning and then be reused until the pipeline is stopped.

A simple example demonstrating the data flow between components of a pipeline is shown in Figure 5.14.

5.4.2 Components

It is the goal of SSI to let developers quickly build pipelines, but hide as many details about the internal data flow as possible. Components offer this level of abstraction. A developer only indicates the kind of streams to be processed (input and/or output) and proceeds on the assumption that the desired streams will be available at run-time. And when building the pipeline he connects the components as if they directly exchanged data. As shown in Figure 5.15 only the top layer, which defines the connection between the processing components, is visible to the developer, while the bottom layer remains hidden.

Pipelines are built up of three basic components denoted as *sensor*, *transformer*, and *consumer*. A sensor can be the source of one or more streams, each provided through a separate *channel*. Most webcams, for instance, also include an in-built microphone to capture sound. In this case, the audio stream will be provided as a second channel in addition to the video channel. A consumer is the counterpart of a sensor. It has one or more input streams, but no output. Transformers are placed in between. They have one ore more input streams and a single output stream. By connecting components in series we can build a pipeline like the one in Figure 5.16.

5.4.3 Synchronization

In Chapter 4 we have discussed in detail the need for multimodal databases and new fusion algorithms that are able to model temporal correlations between different modalities [89, 237, 239, 240, 321, 323]. This, however, requires a proper synchronization of the involved modalities. For audiovisual recordings this is fairly easy to achieve since most video capturing tools allow recording audio alongside with the video stream. But there are few options if the task is to record from other devices, such as eye trackers, motion capture suites, physiological feedback systems, etc. SSI seeks a general solution to synchronize any kind of sensor data without the need for tailored hardware.

A system is *time synchronous* or *in sync* if all parts of the system are operating in synchrony. Transferred to a pipeline it means we have to be able to specify the temporal relation between samples in different branches, even if they originate from individual sensor devices. Only then it becomes possible to make proper multimodal recordings and to combine signals of different

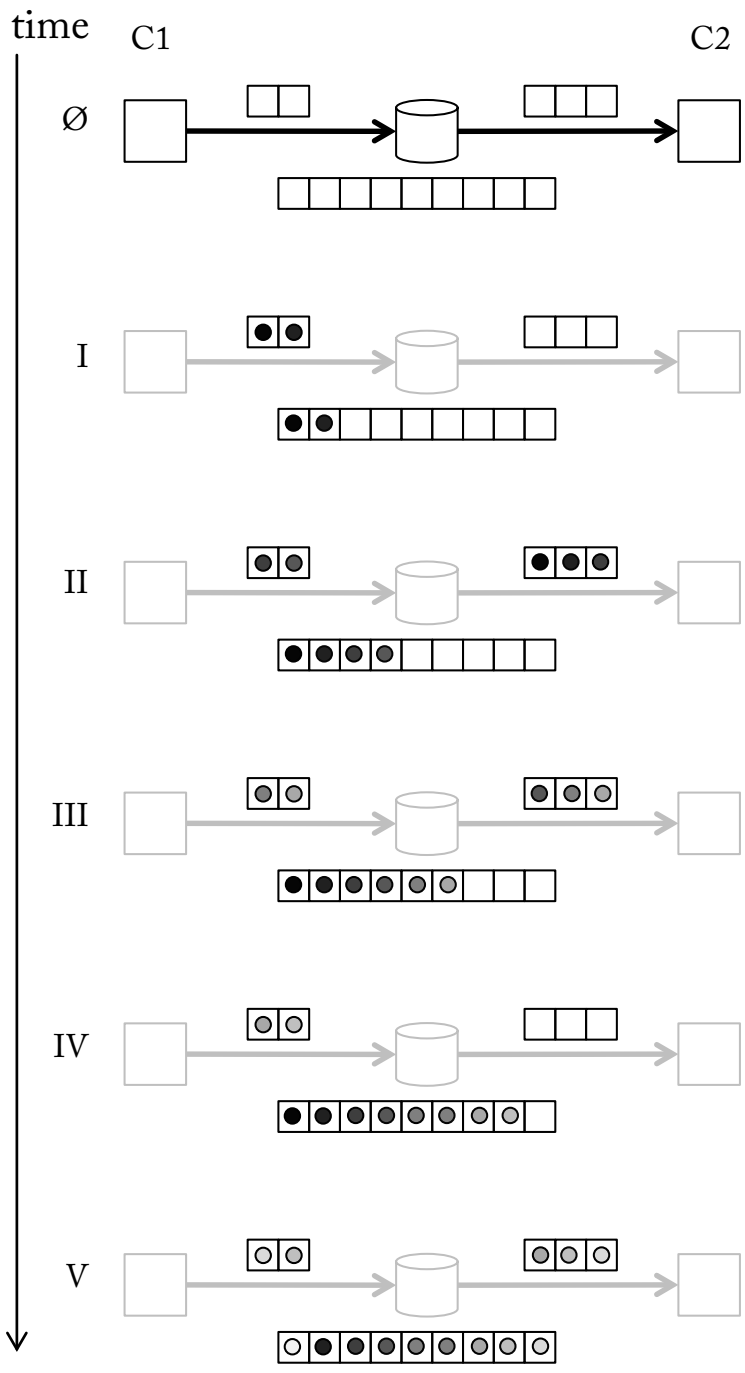


Figure 5.14: The figure depicts the data flow between two components **C1** and **C2** connected through a buffer. Samples are represented by dots in varying gray colours. At each step **C1** writes two samples to the buffer and **C2** sends a request for three samples. Note that although it looks like a synchronous sequence, read and write requests are actually asynchronous operations as **C1** and **C2** run in different threads. Only for the sake of clearness we will treat them in discrete steps triggered by **C1**. In the beginning the buffer is empty and both components are in a waiting state. At step I, **C1** writes two samples to the buffer. Since **C2** requests three samples it is left in waiting state. At step II, **C1** appends another two samples summing up to four samples so that **C2** receives three of them. At step III, **C1** again adds two samples and **C2** receives them together with the sample left over from the previous call. At step IV, **C2** is again in a waiting state, since only two new samples are available and so on.

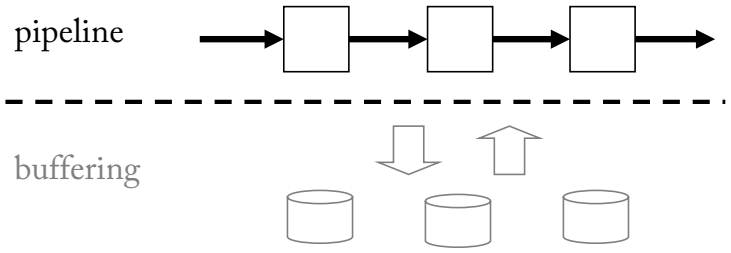


Figure 5.15: In SSI only the top layer, which defines the connection between the processing components, is visible to the developer, while the bottom layer remains hidden.

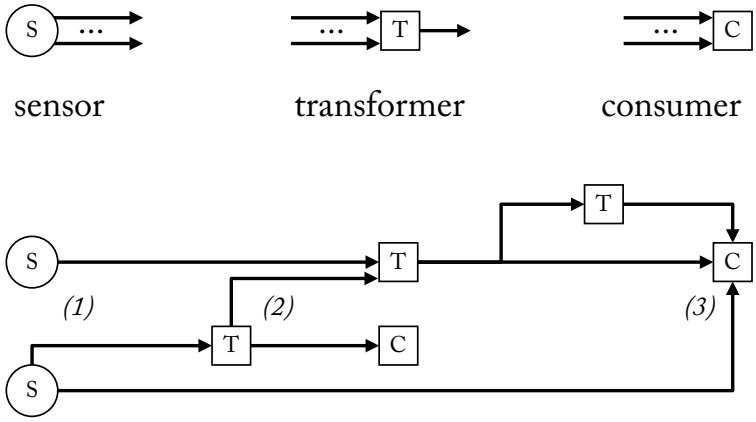


Figure 5.16: Pipelines are built up of three basic components denoted as *sensor*, *transformer*, and *consumer*. A sensor is the source of one or more streams. A transformer receives streams, manipulates them and forwards the result in a single stream. A consumer reads one or more streams, but has no output. By connecting components in series we can build a pipeline like the one in the lower part of the figure. It begins with three streams (1), which are processed along different branches (2), and finally combined by a single component (3).

sources. To keep streams in sync, SSI uses a two-fold strategy: First, it is ensured that the involved sensors start streaming at the same time. Second, in regular intervals it is checked that the number of actually retrieved samples matches the number of expected samples (according to the sample rate).

To achieve the first, SSI hast to make sure all sensor devices are properly connected and a stable data stream has been established. Yet, samples are not pushed into the pipeline until a signal is given to start the pipeline. Only then, buffers get filled. Theoretically, this should guarantee that streams are in sync. Practically, there can still be an offset due to different latencies that it takes for the sensors to capture, convert and deliver the measured values to the computer. However, it is hardly possible to compensate for those differences without using special hardware. Given that these latencies should be rather small ($< 1\ ms$), it is reasonable to ignore them.

Actually, if sensors would now stick precisely to their sample rate, no further treatments were necessary. However, fact is that hardware clocks are imperfect and hence we have to reckon that

the specified sample rates are not kept. Hence, the internal timer of a buffer, which is updated according to an "idealised" sample rate, will suffer from a constant time drift. To see why, let us assume a sensor that is supposed to have a sample rate of 100 Hz, but in fact provides 101 samples every second. During the first 100 s we will receive 10100 samples, which results in a drift of 1 s. And after one hour we will already encounter an offset of more than half a minute. Matching the recording with another signal captured in parallel is no longer possible unless we are able to measure the drift and subtract it out which is impossible if it is non-linear.

Obviously, such inaccuracies will propagate through the pipeline and cause a time drift between branches originated by different sources. The pipeline will be out of sync. To solve this issue we can compare the buffer clock with a global time measure applying a similar strategy that is used in sensor networks. Only, that in our case the propagation time to receive the global time can be neglected since it is directly obtained from the operating system. Let us understand what happens when we adjust the internal clock of a buffer. If the sample rate was lower than expected it means that too few samples were delivered. If we now set the clock of the buffer ahead in time components that are waiting for input will immediately receive the latest sample(s) once again. And this will compensate the loss. Accordingly, if the sample rate was greater than expected it means that we observe a surplus. In this case the buffer is ahead of the system time. If we reset the clock of the buffer, components that are on hold for new input, now will have to wait a little bit longer until the requested samples become available. Practically, this has the effect that a certain number of samples are omitted.

Of course, duplicating and skipping samples changes the propagated streams and may introduce undesired artifacts. However, as long as a sensor works properly we talk about a time drift of few seconds over several hours at the worst. If buffers are regularly synchronised only every now and then a single sample will be duplicated or skipped. Now, what if a sensor stops providing data, e.g. because the connection is lost? In this case updating the buffer clock would cause the latest samples to be sent over and over again. A behaviour which is certainly not desirable. Hence, if for a certain amount of time no samples have been received from a sensor, a buffer will start to fill in default samples, e.g. zero values. Although we still lose stream information at least synchronisation with other signals is kept until the connection to the sensor can be recovered (see Figure 5.17).

So far, we have considered synchronisation at the front of a pipeline. Now let us focus on a transformer, which sits somewhere in between. As long as a transformer receives input at a constant sample rate and outputs at a constant rate this will preserve synchronisation. For example, a transformer that always outputs half as many samples as it received as input, will exactly halve the sample rate. However, it may happen that a transformer works too slow, i.e. is not able to process the incoming stream in real-time. For some while this will be picked up by the buffer it receives the input from. But at some point the requested samples will not be available since the they date too far in the past. Now the pipeline will block. To prevent this

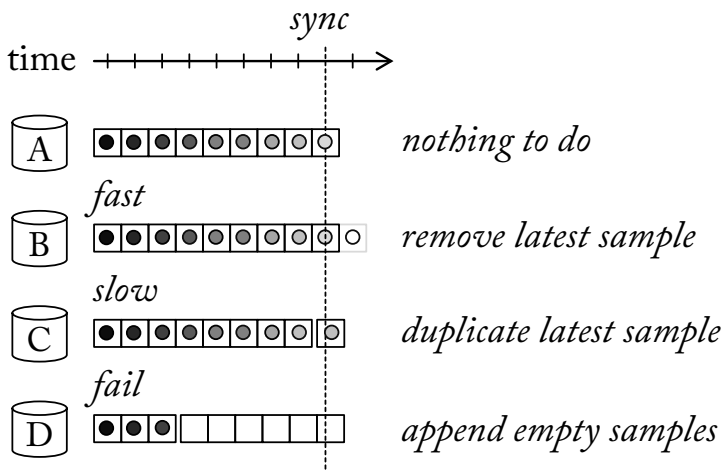


Figure 5.17: In regular intervals the clock of a buffer is synchronised with a global time-stamp. If the clock of a buffer runs fast, i.e. more samples were received as expected according to the sample rate, samples are removed. Likewise, if the buffer has fallen behind samples at the front are duplicated. In case of a sensor fail the buffer is filled with default samples, e.g. zero values.

situation transformers can work asynchronously. A transformer that runs in asynchronous mode does not directly read and write to a buffer. Instead it receives samples from an internal buffer that is always updated with the latest samples from the regular input buffer. This prevents the transformer to fall behind. A second internal buffer provides samples to the regular output buffer according to the expected sample rate and is updated whenever the transformer is able to produce new samples (see Figure 5.18).

5.4.4 Events

So far we have exclusively talked about continuous signals. However, at some point in the pipeline it may not be convenient any more to treat information in form of continuous streams. An utterance, for instance, will neither occur at a constant interval, nor have equal length as it depends on the content of the spoken message. The same is true for the duration of a gesture, which depends on the speed at which it is performed or even shorter spans such as fixations and saccades in the gaze. Also changes in the level of a signal, e.g. a raise in pitch or intensity, may occur suddenly and at a irregular time basis. At this point it makes sense to stick to another representation, which we denote as *events*. An event describes a certain time span, i.e. it has an start and end time, relative to the moment the pipeline was started and in this way are kept in sync with the streams. But in contrast to streams they are not committed to a fixed sample rate, i.e. they do not have to occur at a regular interval. Events may carry meta information, which add further description to the event. For instance, the recognised key word in case of a key word event.

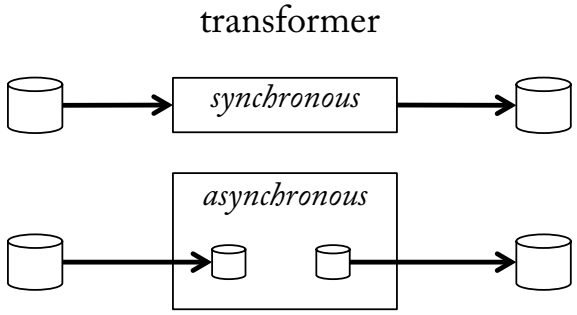


Figure 5.18: A transformer that runs in synchronous mode receives samples directly from the input buffer, manipulates them and writes the result to the output buffer. To not fall behind and block the pipeline it supposed to finish operations in real-time. If this cannot be guaranteed it is run asynchronously. In this case two intermediate buffers ensure that samples can be constantly transferred according to the sample rate. Whenever the transformer has successfully processed data from the internal input buffer it updates the values in the internal output buffer.

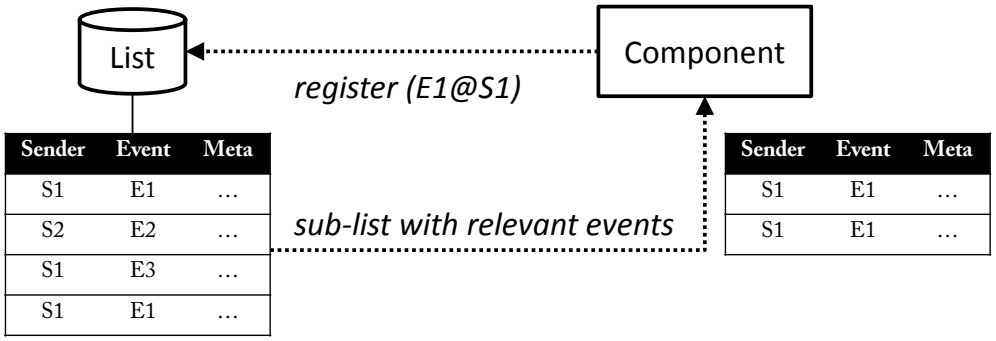


Figure 5.19: Components can register for events filtered by address. The component on the right, for instance, only receives events with name *E1* that are sent by a sender with name *S1*.

Components can send and receive events, and each event has an *event name* and a *sender name* to identify its origin. The two names form the address of the event: *event@sender*. The event address is used to register for a certain type of events. Addresses of different events can be put in row by comma, e.g. *event1,event2@sender1,sender2*. Omitting one side of the address will automatically register all matching events, e.g. *@sender* will deliver all events of the specific sender and a component listening to @ will receive any event. There can be any kind of meta data associated with an event, e.g. an array of numbers, a string, or more complex data types. Events are organised in a global list and in a regular interval forwarded to registered components. A component will be notified how many new events have been added since the last update, but may also access previous events (see Figure 5.19).

Since a consumer does not output a new stream, events can be used to *trigger* when data should be consumed. In this case the consumer does not receive continuous input, but is put in waiting state until the next event becomes available. When a new event occurs, it will be provided with

the stream that corresponds to the time frame described by the event as depicted in Figure 5.20. In this way, processing can be triggered by activity, e.g. apply key word spotting only when voice is detected from the audio. Of course, it is also possible to trigger across different modalities. For instance, activate key word spotting only if the user is looking at certain objects as then we expect him to give commands to manipulate the object.

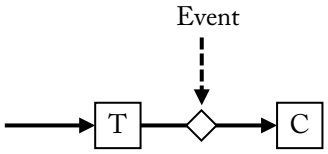


Figure 5.20: Consumer triggered by an event. T=Transformer, C=Consumer

5.4.5 Classification

Machine-aided learning offers an alternative to explicitly programmed instructions. It is especially useful to solve tasks where designing and programming explicit, rule-based algorithms is infeasible. Instead of manually tuning the recognition model, appropriate model parameters are automatically derived after having experienced a learning data set. The quality of a model depends on its ability to generalize accurately on new, unseen examples. In Section 3.2 we have described the basic steps involved in a machine learning task. Section 4.1.1 has dealt with the risk of overfitting and related challenges. SSI supports all steps of a learning task, that is feature extraction, feature selection, model training, and model evaluation. Special focus was put to support both, dynamic and static learning schemes (see Section 3.2.5) as well as various kind of fusion strategies. The trained models can be plugged into a pipeline to apply online classification, which will be explained in detail in the next chapter.

Generally, a *classifier* is a system that maps a vector of feature values onto a single discrete value. In supervised learning the mapping function is inferred from labelled training data. The success of this learning procedure depends on the one hand on the *feature representation*, i.e. how the input sample is represented, and on the other hand on the *learning algorithm*. The two entities are closely connected and one cannot succeed without the other. The learning task itself starts from a set of training samples which encode observations whose category membership is known. A training sample connects a measured quantity, usually represented as a sequence of numbers or a string, with a target. Depending on the learning task targets can be represented as discrete class labels or continuous values. The aggregation of training samples is called a dataset. The learning procedure itself works as follows: a set of training samples is presented to the classification model and used to adjust the internal parameters according to the desired target function. Afterwards, the quality of the model is judged by comparing its predictions on previously unseen samples with the correct class labels.

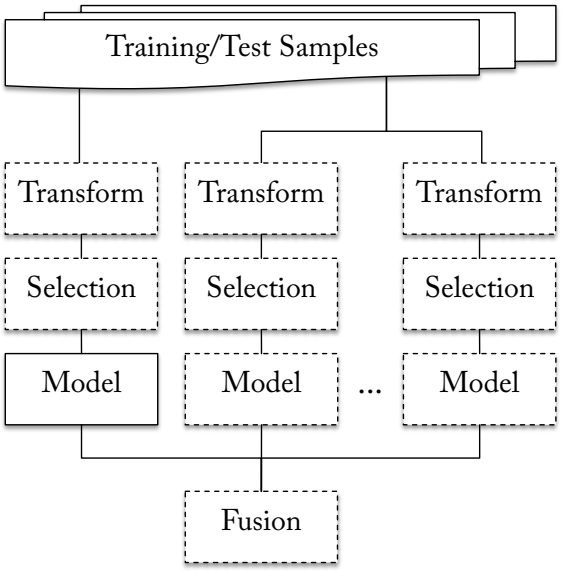


Figure 5.21: Learning in SSI is organised in a hierarchical structure. Blocks in dashed lines are optional.

As illustrated in Figure 5.21 a classifier in SSI is a hierarchical construct that combines the decisions of an ensemble of models in a final fusion step. The samples in the training set can stem from multiple sources, whereby each model is assigned to a single source. However, it is well possible to have different models receive input from the same source. Before samples are handed over to a model they may pass one ore more transformations. Also, if only a subset of the features should participate in the classification process an optional selection step can be inserted to choose relevant features. The learning phase starts with training each of the models individually. Afterwards the fusion algorithm is tuned on the probabilities generated by the individual models. The output of the fusion defines the final decision and is output by the classifier.

5.4.6 Fusion Levels

Combining the predictions of an ensemble of classifiers is one way to fuse information. However, SSI offers several more possibilities. At an early stage two or more streams can be merged into a new stream, which is *data level fusion*. An example for data fusion is *image fusion*, which is the process of registering and combining multiple images from single or multiple imaging modalities to improve the imaging quality and reduce randomness and redundancy. The fusion of medical images has proved to be useful for advancing the clinical reliability of using medical imaging for medical diagnostics and analysis [153]. An example is the fusion of Computer Tomography (CT) and Magnetic resonance (MR) images to highlight anatomical and physiological characteristics of the human body [152]. Early data fusion in SSI is implemented using a transformer with multiple input streams (see Figure 5.22).

Combing multimodal information at *feature level* is another option. It provides more flexibility

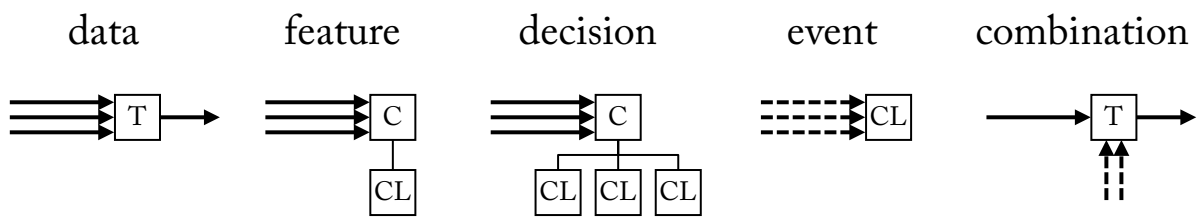


Figure 5.22: Multimodal information can be combined at different stages, ranging from early data fusion to purely event-based fusion, or even a combination of both. T=Transformer, C=Consumer, CL=Classifier

than data fusion since features offer a higher level of abstraction. In feature level fusion features of all modalities are concatenated to a super vector and presented to a classifier. Usually, classifiers are seated in a consumer as depicted in Figure 5.22 and the result of a classification is output as an event. However, if classification is applied on a frame-by-frame basis, it can be replaced by a transformer. In this case, class probabilities are written to a continuous stream. *Decision level fusion* is similar, but individual features sets are classified first and then class probabilities are combined afterwards. Feature and decision level fusion can be implemented using the techniques described in the previous section.

Finally, information can be combined at *event level*. In this case the information to be fused has to be attached to the events, e.g. the result of previous classification steps. Fusing at event level has the advantage that modalities can decide individually when to contribute to the fusion process. If no new events are created from a modality it stays neutral.

5.5 File Formats

Storing the raw and processed data of a pipeline is a key function for building multimodal corpora. Generally, the file format defines how information is encoded for storage on a medium, e.g. a hard disk. Two type of information have to be stored: information describing the properties of a stream and the actual stream data. We distinguish between formats dedicated to a certain signal type and those designed for storage of several different types of data. The latter are referred to as *container* formats. Also, we have the possibility to store information as *binary* or *text*. Binary files can be thought of as a sequence of bytes. It is not possible to read a binary file without an according decoder. The content of a file in text format, on the other hand, can be viewed with any text editor. Note that in effect a text file is still binary, but of a special kind. While it is surely beneficial to use text files due to their readability, binary files have the advantage of compactness.

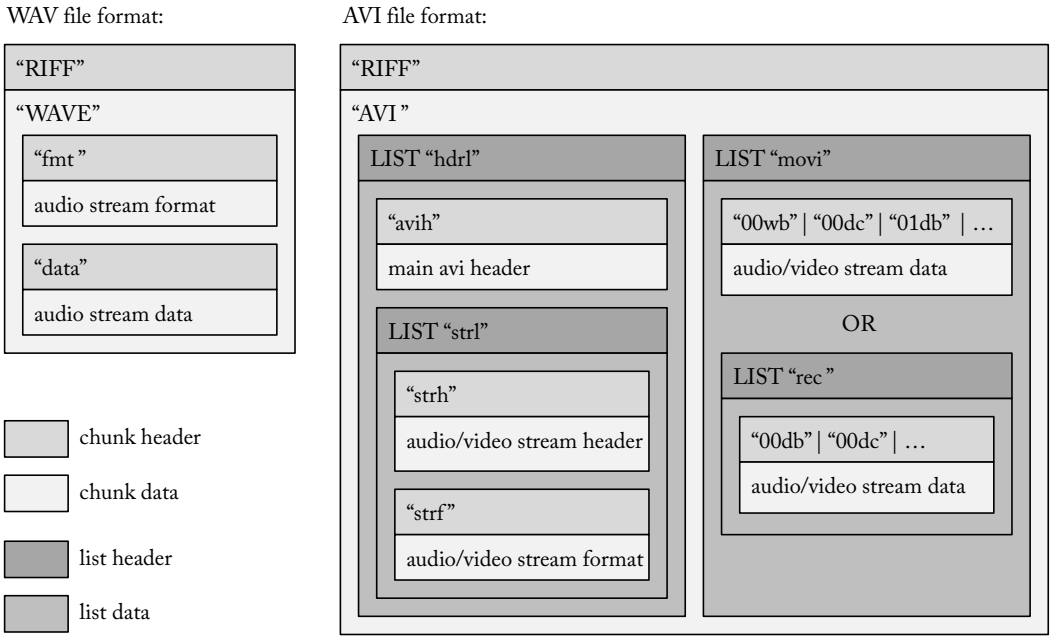


Figure 5.23: WAV and AVI are common standards derived from the generic RIFF container format. RIFF defines chunks consisting of a identifier and some chunk data. Chunks that can contain sub-chunks are called lists. WAV has been developed for storing digital audio bitstreams, AVI can contain contain both audio and video data and allows synchronous audio-with-video playback. A simplified scheme of the two formats is shown above. The mixing of meta information and sample data make it difficult to work with those formats.

5.5.1 RIFF Format

Certain compression formats have been developed to reduce the space a signal would normally require. Among those, *lossless compression* formats allow the exact original data to be reconstructed from the compressed data, whereas *lossy compression* formats discard some of the information, although they usually achieve considerably higher compression rates. Two popular file formats are WAVE or WAV (*Waveform Audio File Format*), a standard for storing audio bitstreams, and AVI (*Audio Video Interleaved*), a multimedia container format that can contain both audio and video data.

Both, WAV[5] and AVI[6], are based on the *Resource Interchange File Format* (RIFF) a container format for saving data in tagged chunks. WAV is specialised in storing audio bitstreams, while with AVI it is possible to combine audio and video data in a single container for synchronous audio-with-video playback. Both file formats consist of a master chunk followed by format and data chunks. A wave file, for instance, begins with a master "`RIFF`" chunk which includes a "`WAVE`" identifier followed by format and data chunks. A format chunk ("`fmt`" identifier) specifies the format of the data, including the sampling rate, the number of channels and few

[5] `http://www-mmsp.ece.mcgill.ca/Documents/AudioFormats/WAVE/WAVE.html`
[6] `http://www.jmcgowan.com/avi.html`

other parameters. A data chunk ("`data`" identifier) contains the sampled data. Although it can be stored as compressed audio, the most common WAV audio format is uncompressed audio in the linear pulse code modulation (LPCM) format. In case of multiple channels (stereo or surround) samples are stored as interlaced, i.e. values for each channel are stored consecutively before advancing to the next sample. A simplified scheme of the WAV and the AVI format is given in Figure 5.23.

5.5.2 CSV and XML

Although WAV and AVI files are widely used standards they are not very convenient in terms of readability. Since meta information and sample data are mixed up in a single file and due to their binary nature, reading the content requires knowledge about their specific structure. To improve compatibility many programs support simpler exchange formats, such as *comma-separated values* (CSV). A CSV file stores tabular data (numbers and text) in plain-text form. It can contain any number of records, separated by line breaks, and each record consists of fields, separated by some character, most commonly a literal comma or tab. The following CSV snippet lists metropolitan areas by population. It contains a header and values are separated by comma.

```
City;Population;Year
Tokyo;37800000;2010
Seoul;25620000;2012
Shanghai;24750000;
Karachi;23500000;2014
Delhi;India;21753486;2011
```

Since all records have an identical sequence of fields CVS is well suited to store data in form of time-series. Unfortunately, there is no standardised way that would allow to store meta information about the signal.

With the rise of the Internet so called markup languages have gained high popularity, in particular XML (*Extensible Markup Language*). XML is based on a set of rules that are both human-readable and machine-readable. An XML document consists of elements which are divided in *markup* and *content*. Markup constructs, so called *tags*, surround an element's content which can be again an element (<tag>content</tag>). *Attributes* are constructs consisting of a name/value pair that exists within a tag (<tag name=value/>). Since XML is not restricted to a limited set of tags it can be easily extended with new tags and attributes. If some tags or attributes are not understand by a parser they are ignored, but the file can still be processed. The following snippet organises the previous example in XML. The name of the city is encoded as an attribute, other fields are encapsulated in tags. Note the overhead of characters introduced by

the markup constructs, which make XML less suited for storing larger time series.

```
<cities>
    <city name="Tokyo">
        <population>37800000</population>
        <year>2010</year>
    </city>
    <city name="Seoul">
        <population>25620000</population>
        <year>2012</year>
    </city>
    ...
</cities>
```

5.5.3 Stream Format

Since there is no standard for storing signals in general, a variety of different data formats exist which are specialised to a certain type of signal. In [281] Schlögel compares 19 data formats developed for storing biomedical signals. As reason for this variety he claims almost every equipment vendor uses their own specification, which in some cases is not even made available to the public. The open source software library for biomedical signal processing (BioSig)[7] supports currently about 50 different data formats. This variety of different formats obstructs interoperability and increases the costs for software development and maintenance.

To combine the advantages of xml (readability and extensibility) with the advantages of a binary format (compactness) and CSV format (compatibility), SSI introduces a two-file format to store streams: an XML file holds general information about the signal and a CSV file contains the actual sample data. Alternatively, sample data can be stored binary to save disk space. Keeping meta information and data in separate places ensures that stream properties are easily accessible, while CSV files or sequences of raw binary values provide an efficient and also highly compatible way of storing the samples.

The following example shows the header file with stream properties encapsulated in an `info` element followed by several `chunk` tags. For each chunk the start position in bytes is stored in the `byte` field and the according number of samples in the `num` field. Beginning and ending of a chunk in seconds are encoded in the fields `from` and `to`. A stream stored as a whole would consist of a single chunk.

```
<?xml version="1.0" ?>
```

[7] http://biosig.sourceforge.net/

```
<stream ssi-v="2">
  <info ftype="ASCII" sr="10.000000" dim="2" byte="4" type="FLOAT"/>
  <chunk from="1.889971" to="2.289971" byte="0" num="4"/>
  <chunk from="16.165067" to="16.465067" byte="77" num="3"/>
  <chunk from="43.151192" to="43.351192" byte="135" num="2"/>
</stream>
```

The data file contains nothing but the pure sample values. Since ASCII was chosen as a file type in the header file, samples are stored row wise with sample values separated by blanks. In binary format, samples are stored in interleaved order. Comments have been added for clarity reasons.

```
-0.059999 0.224001    // start of 1st chunk
0.299954 0.234692
0.290541 0.484024
0.461890 0.362895     // end of 1st chunk
0.388971 0.436113     // start of 2nd chunk
0.328059 -0.000692
0.208171 -0.001738    // end of 2nd chunk
0.476249 -0.002249    // start of 3rd chunk
0.071604 0.385565     // end of 3rd chunk
```

At this point, SSI's strict stream synchronisation again pays off. Since streams have a common starting point and SSI makes sure the exact number of samples are generated in accordance with their sample rates, it is not necessary to introduce additional synchronisation tags. Although streams are stored in separate files, for any point in time matching samples can be determined through all streams. This also holds for data stored in other file formats that SSI supports, such as the WAV and AVI format.

5.5.4 Training Sets

We also want to store data sets needed in learning tasks (see 5.4.5). A famous file format to store training samples is the *Attribute-Relation File Format* (ARFF), developed for use with the Weka machine learning software [138]. An ARFF file is a purely text based format that consists of two distinct sections. The header of the ARFF file contains a list of attributes and their types. Each attribute describes a column in the following data section, which is either a feature value or a nominal value, e.g. a list of target classes.

```
@RELATION arff

@ATTRIBUTE mean_intensity NUMERIC
@ATTRIBUTE mean_pitch NUMERIC
```

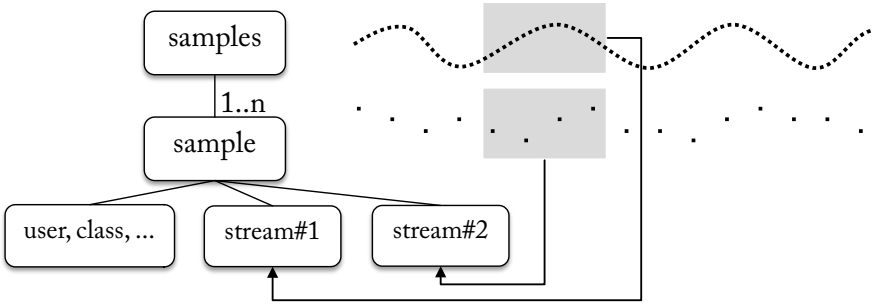


Figure 5.24: A training sample can aggregate data from multiple streams.

```
@ATTRIBUTE class {Happy, Neutral, Sad}

@DATA
75.1, 223.5, Happy
74.9, 243.0, Happy
70.7, 123.2, Sad
74.6, 200.4, Neutral
75.0, 253.6, Happy
75.4, 243.9, Happy
69.6, 153.4, Sad
71.0, 143.4, Sad
74.4, 210.9, Neutral
74.9, 263.1, Happy
```

Although the format is easily understood, it has some shortcomings. Since it is text based the file size quickly grows with the number of attributes and samples. Also, parsing text based content is generally slower than reading in binary data. Its main limitation, however, comes from the fact that it is not suited to store samples of variable length. Hence, its use is restricted to statistical classification tasks in which samples are represented by feature vectors of constant length. To store samples of variable size as they are used in dynamic classification tasks, a more generic file format is needed. Ideally, the format also supports multimodal input, i.e. samples which aggregate data of multiple channels as required in fusion tasks. Hence, the format introduced in SSI seeks a more generic solution allowing training samples to combine several streams of arbitrary length as illustrated in Figure 5.24.

Again, the format is based on multiple files that help separate meta information and sample data. To store sample data it draws on the standard stream format with chunks encapsulating the single samples. This allows samples in the same dataset to vary in length. In case of multimodal data a separate file is used per channel. The paths to the stream files are stored in a header file, which also holds the name of the target classes in the set as well as the names of the users. Grouping samples by user comes in handy if a user independent evaluation is wished. For each category the frequency is stored to provide statistics on the sample distribution over classes and users. Special

flags also specify if the set includes missing data, i.e. (partly) empty samples, and the number of samples that are assigned to a generic garbage class. Samples belonging to the garbage class will not be included in the learning process. At a first glance, it may look odd why we want to keep track of those samples when we can also just remove them from the set. However, if we aim a realistic evaluation we should avoid removing samples but explore ways to deal with them. If a channel fails in a multimodal scenario, for example, we may still have useful information in some of the other channels. And even the percentage of unclassified samples provides a valuable hint regarding the robustness to be expected in a real-life setting.

```
<?xml version="1.0" ?>
<samples ssi-v="3">
  <info ftype="ASCII" size="10" missing="true" garbage="1"/>
  <streams>
    <item path="Stream#0"/>
    <item path="Stream#1"/>
  </streams>
  <classes>
    <item name="Happy" size="2"/>
    <item name="Neutral" size="5"/>
    <item name="Sad" size="2"/>
  </classes>
  <users>
    <item name="Adam" size="5"/>
    <item name="Eve" size="5"/>
  </users>
</samples>
```

In a second file user names and class names are stored for every sample in the set. To save space actual names are replaced by indices. The file also lists magnitude values for use in continuous recognition tasks and time stamps in seconds relative to the start of the pipeline.

```
// <user-id> <class-id> <magnitude> <time>
       0          0          1.00     0.383491 // user Adam, class Happy
       0          0          1.00     0.501764
       0          1          1.00     1.138312
       0          1          1.00     1.327824
       1          1          1.00     1.989954 // user Eve, class Neutral
       1          2          1.00     2.572461
       1          2          1.00     3.387457
       1         -1          1.00     4.213691 // garbage
       1          1          1.00     4.384272
       0          1          1.00     4.989832
```

As mentioned before the actual sample data is transferred to one or more stream files. Each sample is encapsulated in a chunk, which corresponds to an entry in the above file. To keep order, empty chunks (num=0) and garbage samples are kept.

```
<?xml version="1.0" ?>
<stream ssi-v="2">
  <info ftype="ASCII" sr="10.000000" dim="2" byte="4" type="FLOAT"/>
  <chunk from="0.383491" to="0.583491" byte="0" num="2"/>
  <chunk from="0.501764" to="1.001764" byte="39" num="5"/>
  <chunk from="1.138312" to="1.538312" byte="134" num="4"/>
  <chunk from="1.327824" to="1.627824" byte="211" num="6"/>
  <chunk from="1.989954" to="2.389954" byte="345" num="0"/> <!-- empty -->
  <chunk from="2.572461" to="2.872461" byte="345" num="3"/>
  <chunk from="3.387457" to="3.787457" byte="402" num="4"/>
  <chunk from="4.213691" to="4.513691" byte="479" num="3"/>
  <chunk from="4.384272" to="4.784272" byte="537" num="4"/>
  <chunk from="4.989832" to="5.289832" byte="613" num="3"/>
</stream>
```

5.6 Building Components

We want to complete the chapter with a short programming excursion. While more sophisticated examples are to come, the following examples are meant to convey basic programming techniques. SSI is implemented in C++, which has been chosen as programming language due to its bias for systems programming with performance, efficiency and flexibility. As a compiled language, a suitable development environment such as Microsoft Visual Studio is needed to compile SSI code into native machine code. In order to lower the barrier for users not familiar with C++ programming, SSI also offers an XML interface for setting up pipelines with only a text editor. The following samples explain how components are created and plugged into a pipeline in C++. XML pipelines will be introduced at the end of the chapter.

5.6.1 Software Patterns

Before we go into the code, let us point out some software patterns which have been frequently applied.

- **Container Pattern**
  The two most important data structures in SSI are `ssi_stream_t` and `ssi_event_t`. The former to store continuous signal values, the latter to represent discrete event data. Both

are implemented as *container*, which allows them to hold different signal and event types. A stream, for instance, may contain a video frame, an audio chunk or a sequence of gaze points, whereas an event may contain class probabilities, a set of feature values or a string.

- **Interface Pattern**
  SSI components can be classified into few basic classes, such as *sensors*, *filters*, etc. Although implementations of entities of the same class can be very different, they share a common interface that describes which operations can be called on an instance. Any request that matches the interface of an object may also be sent to that object, regardless of its implementation. For instance, components that are meant to extract features implement `IFeature`, which defines a method `transform()` to calculate the features.

- **Factory Method Pattern**
  Components in SSI are identified by a unique name. By applying the factory method pattern it becomes possible to instantiate components by their unique name. In code we can use the following code to create an instance of a class named `"MyComponent"`: `Component c = Factory::Create ("MyComponent")`. In this way, components can be outsourced into dynamic libraries and load at run-time. An important feature, as it allows to establish a plug-in system and to define pipelines in XML.

- **Singleton Pattern**
  To assemble a pipeline SSI features a class `TheFramework`. It is also responsible to start and stop a pipeline and offers functions to request global properties like the current run time. To make it accessible from every component it is implemented as a *singleton* to ensure only one instance of `TheFramework` exists. As access point `Factory` is used, e.g. `Factory::GetFramework()->GetElapsedTime()`.

- **Event Listener Pattern**
  To exchange events between components, a sender can accept one or more listening components: `sender->setEventListener(listener)`. When an event is fired, the sender invokes a predefined method of the listener: `listener->update()`.

5.6.2 Components

To increase readability and reusability SSI follows a component-based development. The function of a component should be a clearly defined task, such as providing access to a sensor device, extracting a set of related features, or processing a certain type of event. Dividing a complex task into a set of isolated and generic subparts helps to maintain the code and reuse it elsewhere. All components derive from the interface `IObject`. In particular they implement two static functions; `GetCreateName()` returns a component's unique name and `Create()` an instance of the

components.

```
// return unique name
static const ssi_char_t *GetCreateName() {
  return "mycomponent";
};
// create an instance
static IObject *Create(const ssi_char_t *file) {
  return new MyComponent(file);
};
```

The class responsible for creating and maintaining components is called `Factory`. To allow individual tuning of components a component can define a set of options by deriving from `OptionList`. The entries of an option list can be stored to an XML file, which makes it possible to adjust them without recompiling the code. The following implementation adds two options, a boolean value and a float value.

```
class Options : public OptionList {
public:
  Options()
    : debug(false), threshold(0.0f) { // set default values
    // define options by name, type and help string
    addOption("debug", &debug, 1, SSI_BOOL, "turn␣on␣debug␣messages");
    addOption("threshold", &thres, 1, SSI_FLOAT, "this␣is␣a␣threshold");
  }
  // option variables to be used by the component
  bool debug;
  float thres;
};
```

We can now register the component and create an instance using `Factory::Create()`. Afterwards, we apply new options and free the memory by calling `Factory::Clear()`. Options are stored in a file `"component.option"`.

```
// register component
Factory::Register(MyComponent::GetCreateName(), MyComponent::Create);
// create instance and set name of option file
MyComponent *c = ssi_create(MyComponent, "component", true);
// override some options
c->getOptions()->threshold = 1.0;
c->getOptions()->debug = true;
// clear memory, options will be stored to file
Factory::Clear ();
```

5.6.3 Sensor

Components can be further categorised according to the task they accomplish. An interface is provided for each of the component types introduced in Section 5.4.2 and depicted in Figure 5.16. The first component we will implement derives from `ISensor`, making it a source from which other components receive input. The interface `IChannel` allows us to define the properties of the stream that will be delivered over a channel (see Section 5.3.3). For the purpose of illustration we want to generate a sine wave with a single cycle per second. The sample rate of the signal should be freely selectable via an option[8].

```
class MySineChannel : public IChannel {
  MySineChannel () {
    // define a 1-dimensional stream of float values sampled at 10 Hz
    ssi_stream_init(stream, 0, 1, sizeof(float), SSI_FLOAT, 10.0);
  }
  // assign a name to the channel
  const ssi_char_t *getName() { return "sine"; };
  // access the stream
  ssi_stream_t getStream () { return stream; };
};
class Options : public OptionList {
public:
  Options()
    : sr(50.0) { // set default value
    // define options by name, type and help string
    addOption("sr", &sr, 1, SSI_DOUBLE, "sample rate of sine wave");
  }
  // option variables to be used by the component
  double sr;
};
```

Negotiation between sensor and pipeline starts with a request in form of a call to `setProvider()`. An instance of type `IProvider` is received, which in the following will act as mediator between the two entities. We call `init()` to pass a pointer to the channel definition and keep a reference for later purpose. Note that we adjust the default sample rate of the channel only right before initialising the channel to apply possible changes by the user.

```
bool MySineSensor::setProvider(const ssi_char_t *name, IProvider *provider) {
```

[8] According to the Nyquist theorem introduced in Section 5.3.2 it would be sufficient to choose a sample rate that is more than twice as large as the maximum frequency of the signal. Since the sample rate of a sine wave with a single cycle per second is 1 Hz a sample rate > 2 Hz would theoretically do. Yet, choosing a higher sample rate will lead to a smoother signal path.

```
    // adjust sample rate from options
  _channel.stream.sr = _options.sr;
  // initialize channel
  provider->init(&_channel);
  // store pointer to provider
  _provider = provider;
  return true;
}
```

Next, `connect()` will be called. It is the time to actually establish a connection to the requested channels. In the example at hand we take the opportunity to initialise the clock of the thread. Since the sampling rate of the channel was set to 50 *Hz*, the elapsed time between two successive samples should be $\frac{1}{50}$ *s*. For convenience we can use the function `setClockHz()`. The function `disconnect()` will be called after the pipeline was stopped. Here, we leave it empty left for the moment.

```
bool MySineSensor::connect() {
  // set thread clock
  setClockHz(_channel.stream.sr);
  return true;
}
bool MySineSensor::disconnect() {
  return true;
}
```

The remaining two functions of `ISensor` are `start()` and `stop()` and meant to start and stop the channel stream. In our case, we take the opportunity to run the embedded thread. The method `clock()` will be looped at regular intervals according to the span set earlier. Within the body of the method we request the current system time and apply the sine function to it. We pass the result over to the provider using `provide()`. The provider feeds the sample into the pipeline. It is also responsible for applying the synchronisation mechanisms described in Section 5.4.3.

```
bool MySineSensor::start() {
  // starts the thread
  return ClockThread::start();
};
void MySineSensor::clock() {
  // calculate next sample value
  float y = sin(2.0f*3.1416f*(ssi_time_ms()/1000.0f));
  // hand value over to provider
  _provider->provide(ssi_pcast(ssi_byte_t, &y), 1);
}
bool MySineSensor::stop() {
```

```
  // stops the thread
  return ClockThread::stop();
};
```

The complete source code is available in the Appendix A.1.1.

5.6.4 Filter

Next we implement the interface `IFilter` to create a filter component. In SSI, a filter is a special case of a transformer which manipulates a stream without changing the sample rate. We still have to specify the other properties of the output stream. Since the filter we implement only converts the values of a stream to their absolute value, sample dimension and the sample type remain unchanged. Hence, we simply forward the properties of the input stream.

```
// determine the number of dimensions of the output stream
ssi_size_t MyAbsFilter::getSampleDimensionOut(ssi_size_t dimension) {
  return dimension;
}
// determine the number of bytes per sample value of the output stream
ssi_size_t MyAbsFilter::getSampleBytesOut(ssi_size_t bytes) {
  return bytes;
}
// determine the type of the sample values of the output stream
ssi_type_t MyAbsFilter::getSampleTypeOut(ssi_type_t type) {
  return type;
}
```

The function `transform()` is called to apply the processing. As parameters an input and an output stream are passed. The output stream is dynamically pre-allocated according to the specified properties. If a transformer is connected to several input sources an array holds the additional streams. Since the SSI stream format is a generic container for any kind of sample values (see Section 5.3.3), we start by casting in- and output streams to their actual sample type (here a `float` pointer). We then iterate over all sample values of the input stream, calculate their absolute values, and assign them to the output stream. The total number of sample values is determined by multiplying the number of dimensions times the number of samples.

```
void MyAbsFilter::transform (ITransformer::info info,
  ssi_stream_t &stream_in,
  ssi_stream_t &stream_out,
  ssi_size_t xtra_stream_in_num,
  ssi_stream_t xtra_stream_in[]) {
```

```
    // pointer to first sample value
  float *src = ssi_pcast (float, stream_in.ptr);
  float *dst = ssi_pcast (float, stream_out.ptr);
    // iterate over samples and calculate absolute values
  for (ssi_size_t i = 0; i < stream_in.num * stream_in.dim; i++) {
    *dst++ = abs (*src++);
  }
}
```

The complete source code is available in the Appendix A.1.2.

5.6.5 Consumer

To complete the pipeline we finally implement a consumer. In contrast to a transformer, a consumer has no output, hence there is no need to specify properties for an output stream. We derive from `IConsumer` and implement the method `consume()`. The following snippet prints the values of a stream on the console. Values are separated by blanks. After every sample a new line is added. We use two nested loops, the outer one iterates over the number of samples, the inner one over their values.

```
void MyPrintConsumer::consume (IConsumer::info consume_info,
  ssi_size_t stream_in_num,
  ssi_stream_t stream_in[]) {
  // pointer to first sample value
  float *src = ssi_pcast(float, stream_in[0].ptr);
  // iterate over samples and print values on console
  for (ssi_size_t i = 0; i < stream_in[0].num; i++) {
    for (ssi_size_t j = 0; j < stream_in[0].dim; j++) {
      printf("%.2f ", *src++);
    }
    printf("\n");
  }
}
```

The complete source code is available in the Appendix A.1.2.

5.6.6 Pipeline

We have now all bits in place to assemble a simple pipeline. We get a reference to `TheFramework`, which will help us to connect components to a pipeline. To place the sine wave generator, we create an instance and add it with `AddProvider()`. We pass the name of the channel (here `"sine"`)

and on access receive an instance of `ITransformable`. We store the pointer to allow other components to connect to the component. Finally, we add the sensor instance with `AddSensor()`. Note that we also change the default rate of 10 *Hz* to 50 *Hz*.

```
MySineSensor *sensor = ssi_create(MySineSensor, 0, true);
sensor->getOptions()->sr = 50.0;
ITransformable *sine_t = frame->AddProvider(sensor, "sine");
frame->AddSensor(sensor);
```

Next we create an instance of the filter, which we connect to the output of the sine generator with `AddTransformer()`. Another instance of *ITransformable* is returned and stored. Beside the input source we have to specify the `frame` size. The frame size defines the length of the processing window. If the value ends on a `s` it is interpreted in units of seconds. In that case the number of samples per window is automatically derived by multiplying the value with the sample rate of the signal. Otherwise, if the trailing `s` is omitted, the value is taken as is. Here, we set the frame size to `"1"` so that the filter is applied as soon as new sample becomes available, which is reasonable due to the rather low sample rate of the input signal.

```
MyAbsFilter *filter = ssi_create(MyAbsFilter, 0, true);
ITransformable *sine_abs_t = \
  frame->AddTransformer(sine_t, filter, "1");
```

Finally, we add an instance of the consumer to print the stream values on the console. We do so by calling `AddConsumer()`. This time, we set a frame size of `"0.01s"`, which translates to 5 samples (0.1 $s \cdot 50$ *Hz*), i.e. in each call the received stream contains 5 samples.

```
  MyPrintConsumer *consumer = ssi_create(MyPrintConsumer, 0, true);
  frame->AddConsumer(sine_abs_t, consumer, "0.1s");
```

To run the pipeline we put all bits in row and call `frame->Start()`. Internally, SSI will now start a thread for each component in the pipeline. When all sensors are connected and a stable connection is established, buffers get filled and processing is kicked off. Execution is stopped with `frame->Stop()`. This will finish processing and disconnect the sensors. To finally release the pipeline and free allocated memory `frame->Clear()` and `Factory::Clear()` should be called.

```
// start pipeline
frame->Start();
// wait for stop signal
frame->Wait();
// stop pipeline
```

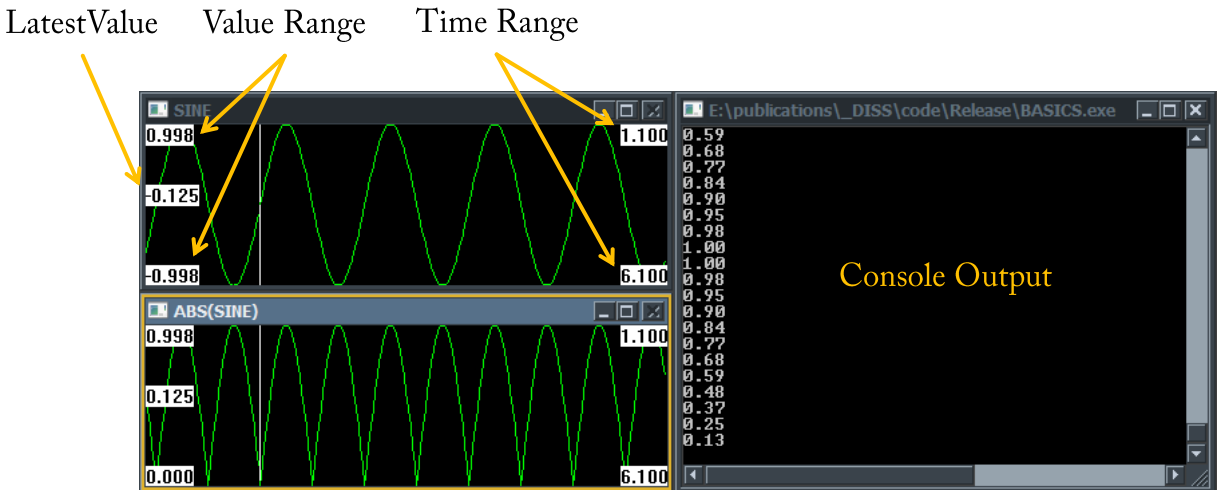


Figure 5.25: Line graphs of the original and processed sine wave (left). Signal values are also printed on the console (right).

```
frame->Stop();
// clean up
frame->Clear();
Factory::Clear();
```

The complete code is available in the Appendix A.1.4. Two instances of `SignalPainter` have been added. `SignalPainter` is part of the core system of SSI and can be used to visualise streams in a line graph. The result is shown in Figure 5.25.

5.6.7 Pipeline in XML

The purpose of SSI framework is twofold. On the one hand, encouraging programmers to implement new functionality and extend the list of available components. On the other hand, addressing users with little or no programming background and give them tools to build and maintain pipelines from available components. To this end, a simple-to-use XML interface comes into play. It allows defining pipelines in pure XML.

In order to use components in XML they are exported to a *dynamic-link library* (dll). Entry point is a function called `Register`, inside which any number of components can be registered (see Section 5.6.2).

```
// entry point for the dll, registered components will be exported
DLLEXP bool Register(ssi::Factory *factory, ...) {
  // register components
  Factory::Register(MySineSensor::GetCreateName(), MySineSensor::Create);
  Factory::Register(MyAbsFilter::GetCreateName(), MyAbsFilter::Create);
```

```
    Factory::Register(MyPrintConsumer::GetCreateName(), MyPrintConsumer::Create);
}
```

To import a dll to an XML pipeline <register> is available. Generally, instances of components are created using <object>. Specialised components are introduced using certain tags. For instance, we use <sensor> to add a sensor. A connection to a channel is established with <provider>. Two attributes are expected: channel is the name of the channel and pin defines a freely selectable identifier, which is used by other components to connect to the stream. Options are directly accessible as attributes. The attribute option is reserved to specify the name of the option file.

```
<sensor create="ssi_sensor_Sensor" sr="50.0">
  <provider channel="sine" pin="sine_t"/>
</sensor>
```

Transformers are marked by <transformer>. Within <input> we determine the input stream for the transformer by setting the according pin. The <output> tag assigns a new pin, which allows following components to connect to the result of the transformation.

```
<transformer create="ssi_filter_MyFilter">
  <input pin="sine_t" frame="1"/>
  <output pin="sine_abs_t"/>
</transformer>
```

To add a consumer we wrap it with <consumer>.

```
<consumer create="ssi_consumer_MyConsumer">
  <input pin="sine_abs_t" frame="0.1s"/>
</consumer>
```

Finally, we wrap everything in an element named <pipeline>, save it to a file and call the interpreter.

```
> xmlpipe.exe <filepath>
```

The complete pipeline is available in the Appendix A.1.4. It will produce the same result as shown in Figure 5.25.

5.6.8 Variables in XML

Sometimes, it is clearer to outsource important options of an XML pipeline to a separate file. In the pipeline we mark those parts with `$(<key>)`. A configuration file then includes statements of the form `<key> = <value>`. For instance, we can assign a variable to the sample rate option.

```
<sensor create="ssi_sensor_Sensor" sr="$(sine:sr)">
  <provider channel="sine" pin="sine_t"/>
</sensor>
```

And define the value in a separate file.

```
sine:sr = 50.0  # sample rate of sine wave
```

To apply the file we call `xmlpipe.exe` with `–config <filepath>`. For convenience, the tool `xmlpipeui.exe` lists variables and allows automated parsing of all keys from a pipeline. Via dropdown one can quickly switch between available pipelines/configuration files and run them right off (see Figure 5.26).

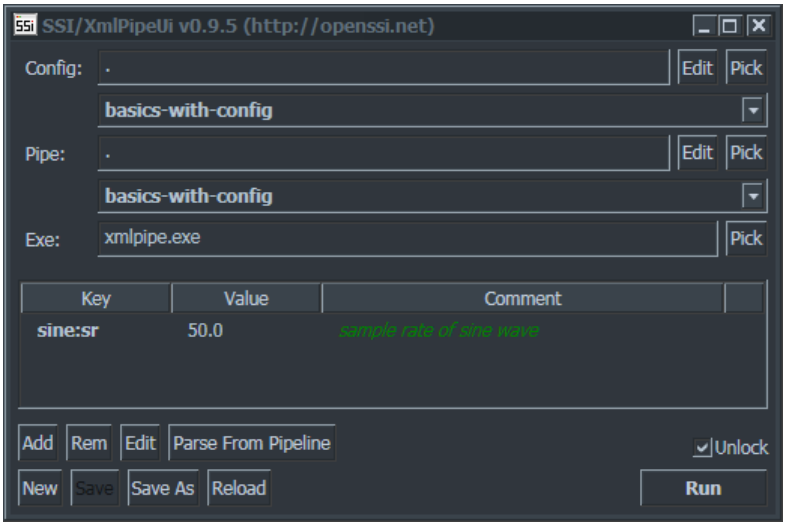


Figure 5.26: Graphical interface to manage options and run pipelines with different configurations.

5.6.9 XML Editor

SSI offers a built-in XML editor which simplifies the task of assembling pipelines. Available components are listed grouped by type. Depending on the type of component that is added, valid XML code is automatically created and inserted. For a selected component the interface displays available options which can be edited in place (see Figure 5.27).

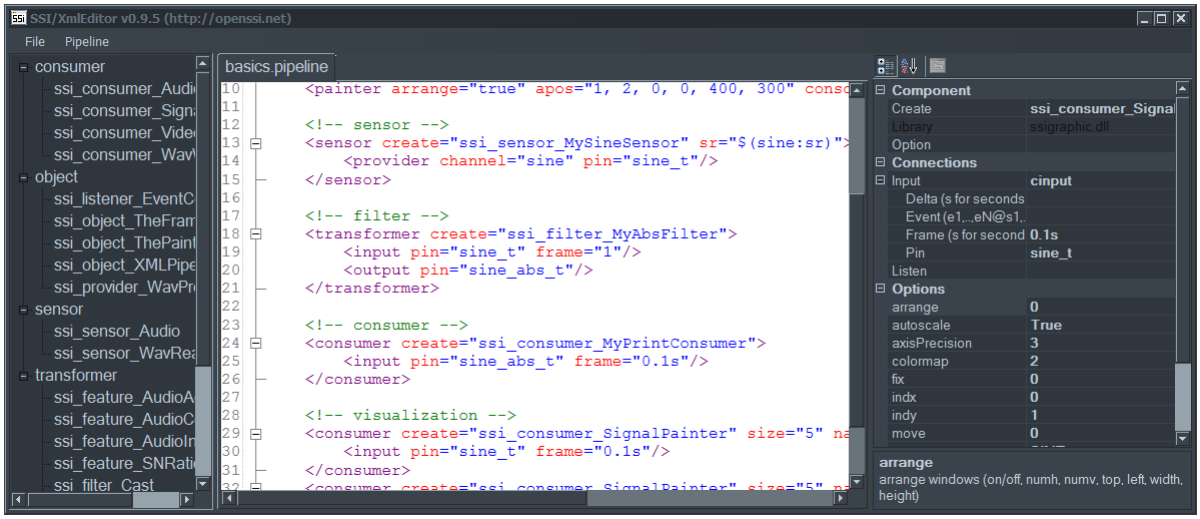


Figure 5.27: SSI's XML editor offers convenient access to available components (left panel). A component's options can be directly accessed and edited in a special sheet (right panel).

Chapter 6

Emotional Speech Recognition

In this chapter we finally want to put hands on SSI. As a concrete example we choose the task of emotional speech recognition. The system we build is basically a clone of EMOVOICE, a real-time system to recognise emotions from acoustic properties of speech [328]. In the course of the example we will reassemble the original system from autonomous components. Once transferred to an SSI pipeline, the modular design will allow us to go beyond the original functionality and improve it by adding a second set of features.

6.1 Basic Recognition System

In Chapter 2 we have emphasised the role of tone colouring in human speech as a carrier of nonverbal information in human communication. As discussed in Section 3.2 these changes can be measured by the means of features derived from pitch contour, intensity level, and sound spectrum. In a training step we link the features to a set of predefined emotional categories. To set up a real-time recognition pipeline, raw signal capturing, feature extraction and classification need to be continuously executed and coordinated. The SSI framework offers tools to accomplish this task. All examples in the text are completely realised in XML and do not require a special building environment, nor programming knowledge.

6.1.1 Audio Sensor

The input to an emotional speech recogniser is spoken content. To capture a user's voice we start by adding an audio source to the pipeline. The component for this purpose is called Audio. It is based on the *Windows Multimedia Device* (MMDevice), which enables audio clients to discover audio endpoint devices, determine their capabilities, and establish a flow of audio data

between the component and an audio endpoint device. Alternatively, SSI provides a component `AsioAudio` that implements the *Audio Stream Input/Output* (ASIO) protocol specified by Steinberg. Since it bypasses the normal audio path through layers of intermediary operating system software it connects directly to the sound card hardware, which reduces latency and offers a way of accessing multiple audio inputs. We choose a sample rate of 16 *kHz*. This is reasonable since the normal voice range of humans is about 500 *Hz* to 2 *kHz*. Options are stored to a file `audio` which will save the audio endpoint device the user has to select when the pipeline is started the first time.

```
<sensor create="ssi_sensor_Audio" option="audio" sr="16000">
  <provider channel="audio" pin="audio_t"/>
</sensor>
```

6.1.2 Filtering and Feature Extraction

The raw audio signal is not yet applicable for emotional speech recognition. It needs to be transformed into a set of features which best characterises emotions. Prior to this the raw sound is sent through a high-pass filter to compensate the high-frequency part that was suppressed during the sound production mechanism of humans. This step is known as pre-emphasis. Accordingly, the responsible component in SSI is called `PreEmphasis`. We set the frame size to `"0.01s"` which translates to 160 samples (0.01·16 000). In other words, the filter is applied to chunks of 160 samples. After the operation is finished the filter waits until the next complete chunk is available. The value of the pre-emphasis filter is usually between 0.9 and 1.0, here we set it to 0.97.

```
<transformer create="ssi_filter_PreEmphasis" k="0.97">
  <input pin="audio_t" frame="0.01s"/>
  <output pin="emph_t"/>
</transformer>
```

Final feature extraction is encapsulated into a single component called `EmoVoiceFeat`. A total of 1451 features are extracted based on pitch, energy, Mel-frequency cepstral coefficients (MFCCs), the short-term frequency spectrum, and the harmonics-to-noise ratio (HNR). For a detailed discussion see [328]. Unlike a filter, which has the same number of input and output samples, a feature reduces all input samples to a single output sample which is the feature vector. Hence, the behaviour of a feature also depends on the length of the input. Extending the frame size allows us to incorporate more information into a single feature vector, but as a side-effect leads to a slower update rate. As a tradeoff it is possible to enlarge the processing window by an additional "delta" factor, while the frame hop remains unchanged. Testing different combinations of frame and delta size can be worthwhile and hence both values are directly set in the pipeline.

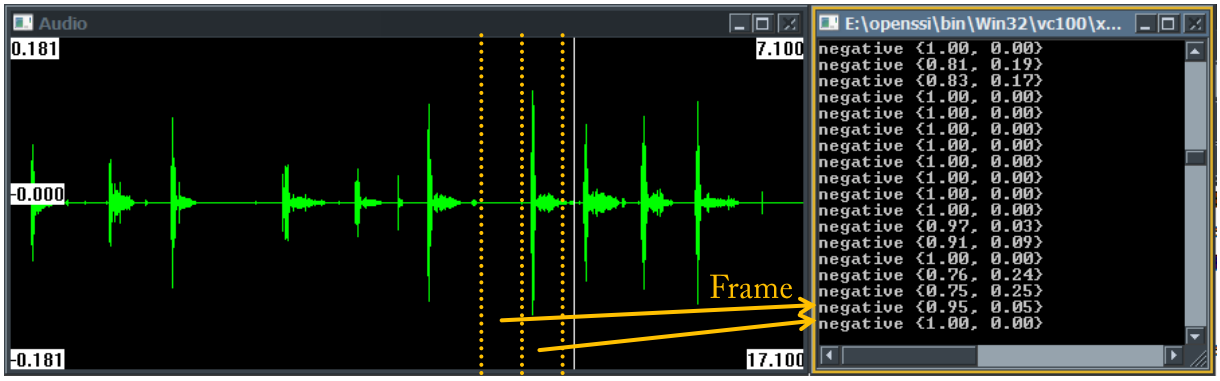


Figure 6.1: A graph displays the captured audio signal. Every 500 *ms* a classification results is output on the console.

For the moment we set `"frame=0.5s"` and `"delta=0.5s"`. Hence the temporal resolution of recognisable changes becomes one second, which should be short enough to cover emotion changes, but still long enough to compute reliable statistical features [327]. After every operation the window will be moved by 0.5 *s* so that the sample rate of the transformed signal is equal to 2 *Hz* ($\frac{1}{0.5s}$). As input we set the output of the pre-emphasis filter.

```
<transformer create="ssi_feature_EmoVoiceFeat">
  <input pin="emph_t" frame="0.5s" delta="0.5s"/>
  <output pin="evfeat_t"/>
</transformer>
```

6.1.3 Classification

To complete the recognition pipeline we add an instance of `Classifier`. It maps the feature stream to a set of class probabilities. The result is provided as an event and forwarded to listening components (see Section 5.4.4). The classification model is loaded from a file. For the moment we will regard the model as a black box. Section 6.4 will be concerned with model training. The frame rate is set to 1 so that a single feature vector is processed at a time. Hence, we will receive a new classification result every half second.

```
<consumer create="ssi_consumer_Classifier" trainer="esr-basic">
  <input pin="evfeat_t" frame="1"/>
</consumer>
```

For the moment the output of our system is limited to the console. This is handy if a recogniser is meant to work in the background but leaves little options to monitor the current state of the system. For debugging we add visual feedback by connecting the audio stream to an instance

of SignalPainter. The following snippet plots the filtered audio signal over the last 10 seconds.

```
<consumer create="ssi_consumer_SignalPainter" size="10">
  <input pin="emph_t" frame="0.1s"/>
</consumer>
```

Now, we have everything in place to run the pipeline. Figure 6.1 shows a screenshot of the pipeline in action. The complete code is available from the appendix (see A.2.1). The system we have built so far replicates the functions of EMOVOICE [328]. However, porting the code to SSI implies several benefits. Despite its compactness (less than 50 lines of XML) it still allows the developer to maintain important properties such as frame rates and file paths.

6.2 Event-based Recognition

The recognition system we have established so far provides a continuous update of the emotional user state. Forcing a frame-wise classification may not be an ideal solution, though. For one thing, it assumes steady speech input, which certainly is not the case if we think of a mutual conversation. For another thing, a fixed segmentation may not align well with the on- and offsets of the emotional content [19]. In [327] Vogt *et al.* have measured recognition performance depending on different segment lengths. Beside a constant frame length of 500 *ms*, they have considered word boundaries as well as whole utterances. Both units are delimited by pauses which facilitates an automated segmentation. Other than a segmentation in chunks of fixed length, the obtained segments are of varying length and do not cover speechless parts. This has the advantage that classification is automatically omitted when no voice signal is available. Apart from that words and sentences as natural linguistic units are likely to coincide with emotional episodes. Vogt *et al.* [327] found whole utterances to be superior to shorter units. Taking this into account we will now extend the recogniser with an activity detector to apply classification only over meaningful signal parts.

6.2.1 Activity Detection

To implement activity detection we employ two more components. Another transformer named AudioActivity which – based on the loudness of the signal – sets values below a threshold to zero (threshold="0.1"). And another consumer named ZeroEventSender which picks up the result and cuts out parts that are non-zero. If a period of activity is detected an event is fired describing its location. Setting mindur="1.0" requests periods to have a minimum length of one second, whereas shorter events will be discarded. The maximal length of segments defaults to 5

seconds. Using options ename and sname we set the event address to active@audio. The event address will allow other components to subscribe for the events sent by this component.

```
<transformer create="ssi_feature_AudioActivity" threshold="0.1">
  <input pin="emph_t" frame="0.03s" delta="0.015s"/>
  <output pin="activity_t"/>
</transformer>
<consumer create="ssi_consumer_ZeroEventSender"
    mindur="1.0" ename="active" sname="audio">
  <input pin="activity_t" frame="0.1s"/>
</consumer>
```

6.2.2 Event-based Classification

So far, we have fed the classifier with a continuous stream of chunks of fixed size. To put the classifier – or more generally any consuming component – into a triggered mode we replace the frame size with the event address of the relevant event(s), here "activity@audio". A consumer in triggered mode waits until an event is reported and then receives the stream chunk that corresponds to the time span defined by the event. However, this also implies another change. So far feature vectors were calculated every half second, producing a continuous stream of samples, which we buffered in a new stream. Now, audio chunks are of variable length as they depend on the frequency and length of asynchronous events. Hence, it is no longer possible to precalculate feature vectors at a fixed sample rate. In such situations, we need to calculate the features right before stream chunks are passed over to the receiving component. To this end, we move the feature component inside the <input> tag to modify the input stream before it is handed over to the consumer.

```
<consumer create="ssi_consumer_Classifier" trainer="esr-event">
  <input pin="emph_t" listen="active@audio">
    <transformer create="ssi_feature_EmoVoiceFeat"/>
  </input>
</consumer>
```

Figure 6.2 concludes the section with a screenshot of the new pipeline. Graphs showing the activity signal and the last triggered audio chunk are added. Classification events are listed in a separate window. The complete code is available in the appendix (see A.2.2).

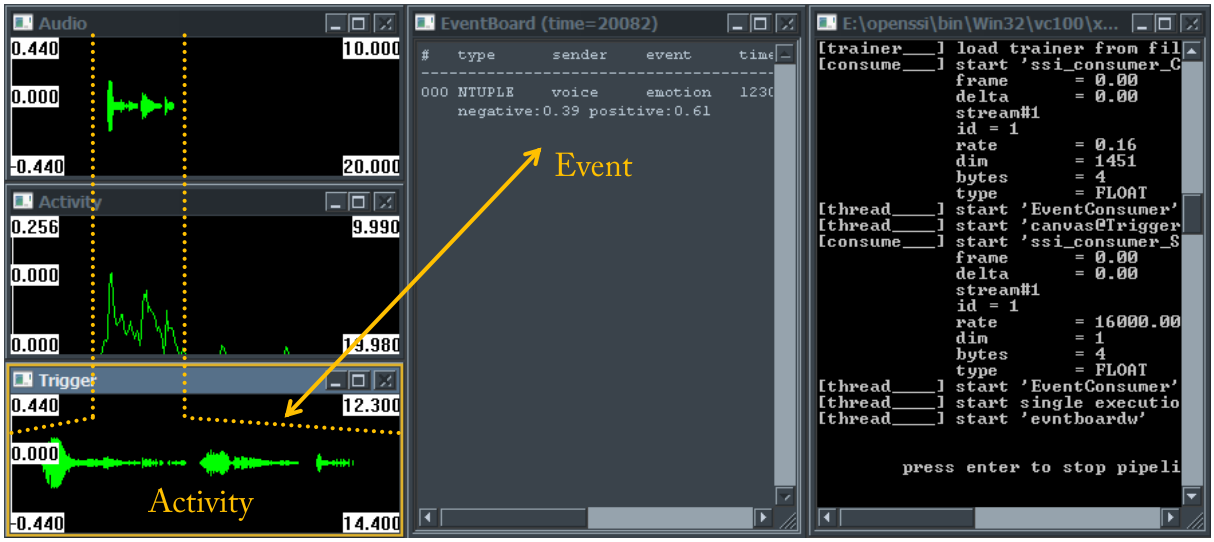


Figure 6.2: The two graphs on the top left display continuous plots of the raw audio and the activity signal within the last 20 seconds. The bottom graph shows an excerpt of the audio signal for the last activity period. The window in the middle collects classification results over the last 10 seconds.

6.3 Data Acquisition

A very sensible part of any recognition system is the classification model. It maps the extracted feature samples to the set of target classes. To automatically learn a generic and robust mapping, we need to provide training data (see Section 3.2.5). The training data has to be carefully selected in terms of naturalness and relevance to the final application as otherwise classification may not lead to reasonable results. We have discussed the various challenges involved in creating appropriate databases in Section 4.1. The possibilities SSI offers for this purpose are manifold. SSI pipelines are lightweight and therefore meant to run in the background. A wizard application may run in parallel and elicit desired user behaviour. Synchronized recordings from multiple devices are supported. Possibly, recordings can be distributed over several computers connected in a network. The captured streams can be processed and stored at various levels of processing. In the following we set up a pipeline that allows a user to record his individual training corpus. The experiment can be repeated as often as needed until a sufficient number of training samples has been collected.

6.3.1 Emotion Elicitation

Providing users with a tool to collect their own training data in a simple and fast fashion bears several advantages. Varying environmental factors such as microphone, background noise, etc., but also speaker characteristics such gender, age etc., make it difficult to build generic models. In particular, since state-of-the-art technology is still not flexible enough to properly cope with

PositiveActive	NegativeActive
I feel amazingly good today!	This is so unfair!
It would really take something to stop me now!	That's dangerous what you're doing, stop it!
You won't believe it, I got the new job!	You really get on my nerves.
PositivePassive	**NegativePassive**
I think my life is beautiful.	I feel rather sluggish now.
I like listening to flowing water in the mountains.	My life is so tiresome.
My life is completely under control.	I just can't make up my mind.

Neutral
Mammals are warm-blooded animals.
The museum is at the end of the road.
No one knows where Mozart is buried.

Figure 6.3: Examples for emotional stimuli sentences inspired by the Velten mood induction technique [320].

these factors. Hence, models trained on existing databases may only be applicable in specific conditions. This also counts for the emotional content that is captured in those databases. In particular, non-prototypical emotions are usually inadequately represented. Hence, a recognition system trained on data specifically collected for a certain task can be expected to yield considerably higher accuracy than a general recognition system.

The method suggested by Vogt *et al.* [328] is inspired by the Velten mood induction technique [320]. It was also adopted by Wilting *et al.* [344], where subjects had to read out loud a set of emotional sentences that should set them into the desired emotional state. Figure 6.3 gives some example sentences for the classes *PositiveActive*, *PositivePassive*, *NegativePassive*, *NegativeActive*, and *Neutral*. The classes have been chosen to cover the quadrants of an *activation-evaluation space* [59]. Neutral has been added to represent the center (see Section 2.4.2). Yet, developers are encouraged to adjust the selection of sentences according to the topics of their application. Regarding the naturalness of the elicited emotions it should be noted that this method does not yield truly spontaneous emotions. Nevertheless, the results can be regarded as semi-acted and still represent a realistic problem because the speakers are usually no professional actors and do not produce full-blown or prototypical emotions as professional actors would do.

6.3.2 Stimuli Presentation

To set up Velten's mood induction technique and similar instruction based experiments it is useful to have a communication layer to present textual or media-based context to the user. As a generic and powerful solution SSI offers the possibility to show web pages in a browser window. The displayed page can be altered via events which contain a new URL address or a path to a local

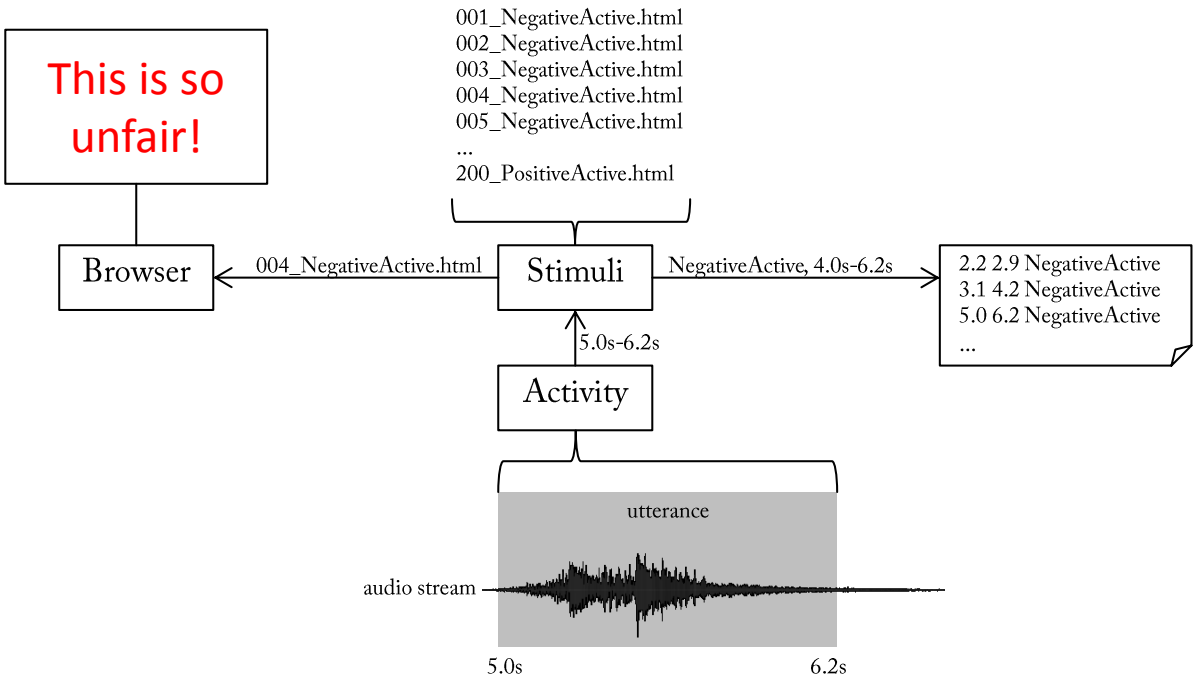


Figure 6.4: Scheme of the training pipeline: when an utterance is detected an event is forwarded to the `Stimuli` component. It stores start and end times and a class label that describes the currently displayed content. Then an event with the next page name is then sent to the `Browser` and a new sentence is presented to the user.

file. That way, a stimuli can be presented until a certain event occurs, e.g. an utterance is detected.

```
<object create="ssi_object_Browser" pos="400,0,400,600">
  <listen address="url@stimuli"/>
</object>
```

We can see that the browser is waiting for events with address `url@stimuli`. We now add a component `Stimuli` which will generate these events. When the component is started it reads filenames from a folder and displays them in the browser. In the concrete case, the files contain the Velten sentences, which are presented to the user who is supposed to read them out loud. The files are grouped by emotional content to make it easier for the user to adapt to the desired mood. When an utterance of the user has been detected start and end time are stored. The component also tries to extract a class label which is saved, too[1]. Then, the file name of the next page is sent to the browser.

```
<object create="ssi_object_Stimuli" sname="stimuli" ename="url"
    folder="velten"
    annoPath="esr_$(date)">
```

[1]Basically, the file extension is removed as well as any non-literals which are needed to keep files in order so that for instance `001_NegativeActive.html` becomes `NegativeActive`.

```
  <listen address="active@audio"/>
</object>
```

To avoid that previous annotations are accidentally overwritten, a time-stamp is added to the name of the dataset. The variable `$(date)` is available for such cause (see Section 5.6.8). Start and endpoints in the file are given in seconds.

```
34.74 37.47 NegativeActive
40.23 42.00 NegativeActive
44.58 47.40 NegativePassive
50.28 52.23 NegativePassive
```

A scheme of the stimuli presentation is given in Figure 6.4.

6.3.3 Stream Recording

Finally, we need to store the audio stream, too. We add an instance of `WavWriter` and connect it to the pin of the preprocessed audio. Again, we use `$(date)` to create a unique file name. It will also help to group files generated within the same recording session.

```
<consumer create="ssi_consumer_WavWriter" path="esr_$(date)">
  <input pin="emph_t" frame="0.1s"/>
</consumer>
```

With only few lines of code we have set up a recording pipeline that not only stores the sensor input, but also displays instructions to the user and creates an automated annotation. The complete code is available in the appendix (see A.2.3).

6.3.4 Creating the Training Set

Running a recording session supplies the ingredients to create a training set. Yet, we need to join annotation and data streams into a dataset of the format described in Section 5.5.4. To this end, SSI offers a graphical tool to visualise a recorded session (see Figure 6.5). Recorded signals are displayed as line graphs and annotation segments as blocks tagged with class labels. Recordings can be replayed to check if the content matches the annotation. If necessary, annotation segments can be added, moved, resized or renamed. Finally, the interface allows creating a training set from an annotation and one or more signal streams. For each segment in the annotation a sample is generated and added to the set.

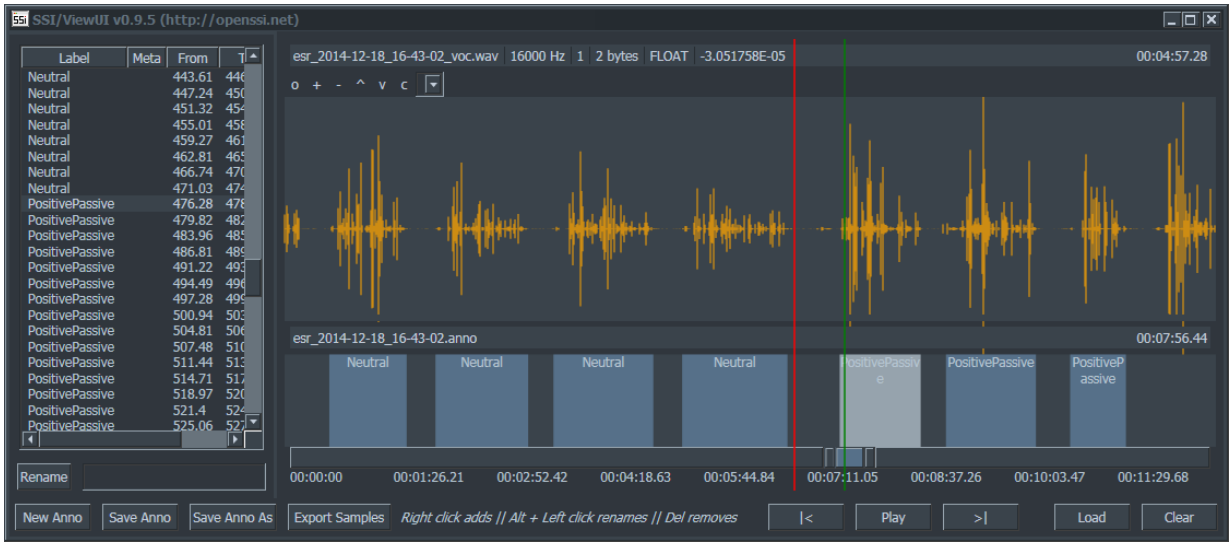


Figure 6.5: A graphical tool helps to review and playback recorded sessions. Streams are displayed as line graphs (top). Annotation tracks can be added or edited (bottom). All segments of the currently selected annotation track are listed in a table (left). Finally, a training set can be automatically extracted from an annotation and one or more signal streams.

In the following we use data from three sessions holding a total of 606 samples more or less equally distributed among the five classes. The audio stream is linked twice, which will allow us to extract two different feature sets. The header of the dataset of the first session looks as follows:

```
<samples ssi-v="3">
  <info ftype="BINARY" size="203" missing="false" garbage="0"/>
  <streams>
    <item path="esr_2014-12-18_12-43-19.samples.#0"/> <!-- audio -->
    <item path="esr_2014-12-18_12-43-19.samples.#0"/> <!-- audio -->
  </streams>
  <classes>
    <item name="NegativeActive" size="41"/>
    <item name="NegativePassive" size="40"/>
    <item name="Neutral" size="42"/>
    <item name="PositivePassive" size="40"/>
    <item name="PositiveActive" size="40"/>
  </classes>
  <users>
    <item name="Johannes" size="203"/>
  </users>
</samples>
```

6.4 Model Training

Once, we have extracted the training samples we can train a classification model. Training is basically a two step procedure. First, a template is defined which specifies the training data and the involved classifiers. Optionally, we can bind additional feature extraction steps or set selections to narrow training to a subset of features. During a training step, the template is converted to a valid classification model to be used in a pipeline (see Section 6.1).

6.4.1 Creating a Template

To create the template we connect it to the datasets we have recorded. The samples in the dataset will be used in the learning phase to train the model. We also choose the classification model that we desire to train, here Support Vector Machines (SVM), and specify the index of the stream. By setting it to 0 we choose the first audio stream. Yet, there is one more issue that deserves our attention.

So far feature extraction has been assumed as a separate step to be accomplished before the training takes place. The samples in the training set, however, have been extracted from the pre-processed audio stream. This means we cannot use them straight away, but need to calculate features first. One possibility would be to convert the dataset and use the transformed set for training. In this case extracting the appropriate feature set is left to the responsibility of the pipeline, i.e. if we decide to switch feature sets, we need to adapt the pipeline accordingly. Although feasible there is smarter way of doing it. Namely, to bind the feature extraction to the model. This way we assure that a model gets the features it needs, independent of the pipeline. The following template connects the three datasets with the EMOVOICE feature set and an SVM classifier.

```
<trainer ssi-v="5">
  <info trained="false"/>
  <samples n_streams="2">
    <item path="esr_2014-12-18_12-43-19"/>
    <item path="esr_2014-12-18_16-43-02"/>
    <item path="esr_2014-12-18_16-56-11"/>
  </samples>
  <transform>
    <item create="ssi_feature_EmoVoiceFeat" stream="0"/>
  </transform>
  <model create="ssi_model_SVM" stream="0"/>
</trainer>
```

6.4.2 Training

To accomplish learning we use the tool `xmltrain.exe`, which takes the template and converts it to a trained model. Additional dll files are referenced via option.

```
> xmltrain.exe -dlls ssiemovoice.dll -out esr esr-tmp
```

The header of a trained model includes information about the input streams as well as class names and user names. Finally, it includes the path to the trained model which stores the learnt model parameters.

```
<trainer ssi-v="5">
  <info trained="true" />
  <streams>
    <item byte="4" dim="1" sr="16000.000000" type="FLOAT"/>
    <item byte="4" dim="1" sr="16000.000000" type="FLOAT"/>
  </streams>
  <classes>
    <item name="NegativeActive"/>
    <item name="NegativePassive"/>
    <item name="Neutral"/>
    <item name="PositivePassive"/>
    <item name="PositiveActive"/>
  </classes>
  <users>
    <item name="Johannes"/>
  </users>
  <transform>
    <item create="ssi_feature_EmoVoiceFeat" stream="0"/>
  </transform>
  <model create="ssi_model_SVM" stream="0"/>
</trainer>
```

6.4.3 Evaluation

Obviously, we want to know how well a model actually performs on the dataset. For this purpose, the tool allows evaluating a template, e.g. using a k-fold cross validation.

```
xmltrain.exe -dlls ssiemovoice.dll -eval 0 -kfolds 2 esr-tmp
```

As output we get a confusion matrix as well as class-wise recognition accuracy and the derived unweighted and weighted accuracy (UA | WA).

```
#classes:        5
#total:          606
#classified:     606
#unclassified:   0
  NegativeActive:     92     1     2    20     7   -->   75.41%
NegativePassive:       4    95     6    14     1   -->   79.17%
         Neutral:      2    14    90    13     5   -->   72.58%
PositivePassive:      18    12    30    50    10   -->   41.67%
  PositiveActive:     45     2     5    38    30   -->   25.00%
                                                 =>   58.76%  |  58.91%
```

As we see, the overall recognition rate is near 60%. Although three times higher than chance level we may still expect many false detections. The confusion matrix, however, reveals that three of the five classes yield an accuracy greater than 70%. Most errors occur within the positive samples which are often mistaken with their negative counterparts. Obviously, the chosen features are not well suited be distinguished along the valence axis. Switching to another feature set might be worth a shot.

6.5 Advanced Recognition System

The modular design of SSI allows us to easily extend our basic recognition system. In the following we will demonstrate this by testing another feature approach and fuse decisions from both feature sets. We will also switch to incremental classification and introduce an interpolation step to fill spots of silence. Finally, we will implement an interface to grant external applications access to the processed data and the final classification result.

6.5.1 Adding Another Feature Set

The Praat software[2] is a tool for the analysis of speech in phonetics. Unlike EMOVOICE it is not meant to produce a very large number of statistical features which are difficult to interpret. For instance, the EMOVOICE feature `mfcc_sma_de[1]_meanPeakDist` encodes the mean peak distance of the first MFCC coefficient (see Section 3.2.4). Even if it turns out to be an useful feature, it will difficult to explain why. Praat, on the other hand, includes algorithms to retrieve few, but meaningful measurements of voice quality, such as the fraction of unvoiced frames or

[2] http://www.praat.org/

local jitter/shimmer. SSI comes with a wrapper to run Praat scripts within a pipeline. To test the performance of the Praat features, we only have to replace the EMOVOICE features with the according component.

```
<transform>
  <item create="ssi_transformer_PraatVoiceReportT" stream="0">
</transform>
```

The new feature set includes only 27 features (rather small compared to 1451 EMOVOICE features). We repeat training and receive a new confusion matrix.

```
 NegativeActive:  73   8   1  16  24  -> 59.84%
NegativePassive:   1  97  10  11   1  -> 80.83%
        Neutral:   4  13  77  27   3  -> 62.10%
PositivePassive:  14  14  26  43  23  -> 35.83%
 PositiveActive:  20   5   9  22  64  -> 53.33%
                                     => 58.39% | 58.42%
```

We see that overall recognition performance has not changed much. This might be disappointing at a first glance. However, comparing class-wise accuracy results reveals an interesting detail. With the Praat set `PositiveActive` is detected twice as accurately as with the EMOVOICE set. This raises hopes that results will improve if we are able to combine the best of both approaches. For this purpose, SSI offers a bunch of well-established fusion methods. An overview can be found in [332]. To apply a fusion rule we add it to the template. We also include both feature extractors, each dedicated to a stream, and two models. During training they will be turned into two individual classifiers whose probabilities are combined using the specified fusion method. Here, we choose `SumRule`, which forms a final decision by summing individual results.

```
<transform>
  <item create="ssi_feature_EmoVoiceFeat" stream="0"/>
  <item create="ssi_transformer_PraatVoiceReportT" stream="1"/>
</transform>
<fusion create="ssi_fusion_SumRule">
  <models>
    <item create="ssi_model_SVM" stream="0"/>
    <item create="ssi_model_SVM" stream="1"/>
  </models>
</fusion>
```

In fact, combining the two sets yields an improvement of more than 5% compared to the single decisions.

```
 NegativeActive:   98   3   1  15   5  -->  80.33%
NegativePassive:    3 102   5   9   1  -->  85.00%
        Neutral:    1   8  93  16   6  -->  75.00%
PositivePassive:   16  11  30  54   9  -->  45.00%
 PositiveActive:   34   5   7  27  47  -->  39.17%
                                        =>  64.90% | 65.02%
```

Only few minor changes are necessary to use the new model with the emotional speech recognizer of Section 6.2. Changing the path to the new model and forwarding a second copy of the audio stream to the classifier. We also remove the feature extraction step, as it has become an integral part of the model.

```
<consumer create="ssi_consumer_Classifier" trainer="esr-fusion">
  <input pin="emph_t" listen="active@audio"/>
  <xinput>
    <input pin="emph_t"/> <!-- add a second copy of the audio stream -->
  </xinput>
</consumer>
```

6.5.2 Incremental Recognition

So far our recognition system outputs a decision only when speech activity is detected. This means we get no decisions during periods of silence. However, the longer the pause lasts the less likely is it that the last detected user state is still valid. If then speech input is detected again a sudden adaptation to a new decision might be necessary. Also, in the current implementation an event is fired only when the end of an activity period is detected. This has the consequence that a decision will be available not before the end of an utterance. This may delay the system response by several seconds and lead to slow system reaction. Finally, it entails that a single false detection can cause a completely inappropriate system behaviour that will not be corrected until another utterance occurs.

A solution to this shortcoming is to produce decisions earlier and with higher frequency. In Section 4.3.4 we have discussed incremental recognition as a technique to produce a decision as early as possible. In our case we can achieve this by invoking classification already before the end of an utterance is detected. For instance, we could always produce a decision over the first second and then after another second using the last two seconds as input and so on (see Figure 6.6). We can implement this behaviour fairly easy using the `incdur` option of the trigger:

```
<consumer create="ssi_consumer_ZeroEventSender"
```

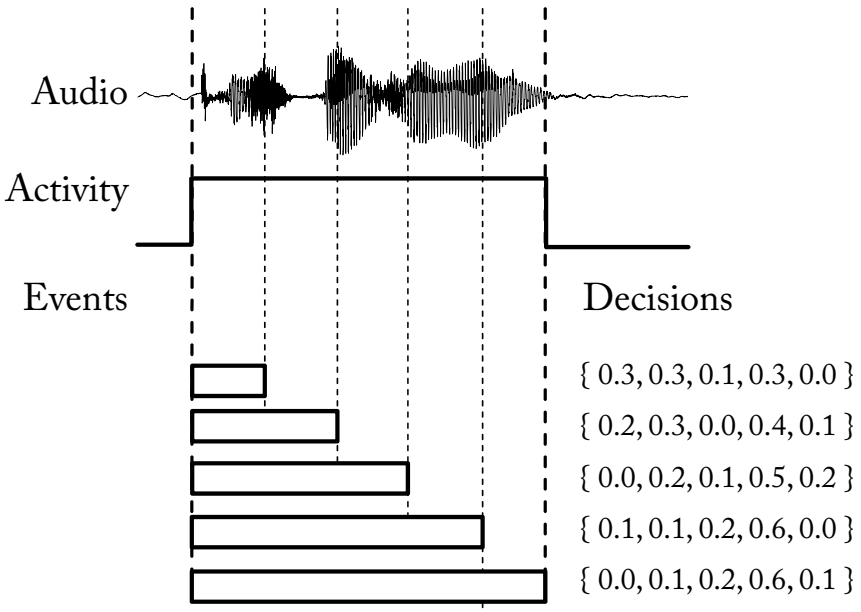


Figure 6.6: To implement an incremental recognition approach a single utterance is segmented into a list of events of increased duration. Events are fired as soon as possible and cause the classifier to output intermediate decisions.

```
    mindur="1.0" incdur="1.0" ename="active" sname="audio">
  <input pin="activity_t" frame="0.1s"/>
</consumer>
```

This only leaves us with the problem that no decisions will be available outside periods of no activity. Also, false detections will still cause an immediate change of the system state, although there is a better chance it will be quickly replaced by a correct prediction. Nevertheless, we can try to decrease the influence of false detections by interposing some sort of interpolation. As a side effect this will also allow us to predict into the future until new confidence becomes available. To achieve this we introduce an intermediate system state and instead of forwarding each recognition result as is, it will only contribute relatively to this overall state. As depicted in Figure 6.7 the final decision is continuously updated on a regular time base to provide a result irrespective of whether new confidence is available or not, e.g. by constantly decreasing the probability until a neutral state is reached. Likewise, the likelihood for a class does not immediately jump to a higher value, but moves stepwise in a new direction. Although this again slows down system reaction, false detections do not have an immediate negative effect[3].

The component that implements the proposed interpolation is called `DecisionSmoother`. It takes decision events and outputs a smoothed decision at a regular time basis, here 100 *ms*.

[3]To speed up system reaction the algorithm can be adopted so that progression becomes faster the more probability for a class is observed. In fact, an advanced version of this rather basic interpolation method will be introduced in Section 7.2 to combine multimodal information.

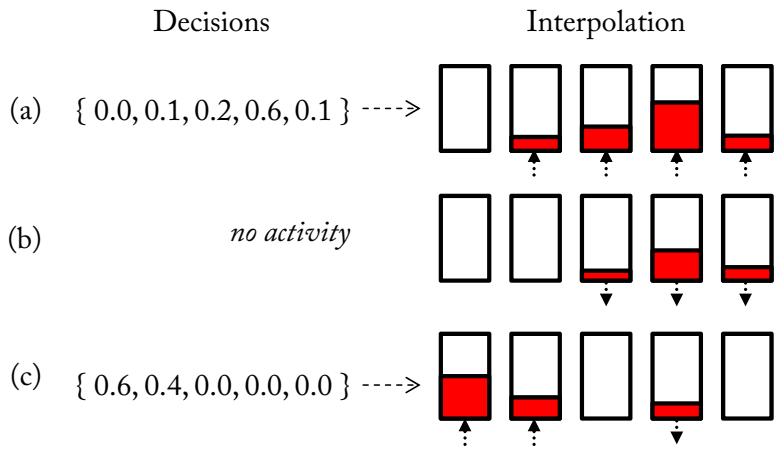


Figure 6.7: To achieve a smoother adaptation of the system reaction and reduce the effect of false detections an intermediate interpolation step can be added. By default classes are set to a neutral position, e.g. 0 probability. (a) A new decision causes the probabilities to grow towards the detected values. (b) Even when no new input is detected likelihoods are updated at a regular interval and decrease towards a neutral state over time. (c) When new activity is detected some of the class probabilities may raise again, while others continue to decrease.

```
<object create="ssi_listener_VectorFusion" dimension="5" update_ms="100">
  <listen address="emotion@voice"/>
</object>
```

6.5.3 Interfacing External Applications

Finally, we want to share processed information with external applications. To interface external applications we need to establish an inter-process communication flow. A very flexible and powerful concept are *network sockets*. Sockets offer a platform independent channel to communicate with applications possibly running on other machines in the network. In a computer network, sockets define the endpoints between which data flow takes place. A socket is represented by a unique *socket address*, which is a combination of an *IP address* and a *port number*. To exchange messages a communication protocol is needed, usually UDP (*User Datagram Protocol*) for connectionless transmission or TCP (*Transmission Control Protocol)* for connection-oriented transmittion. Both protocols are available in SSI.

To forward an SSI stream the consumer `SocketWriter` is available. It wraps incoming frames into UDP or TCP packets and sends them to the specified socket address. Sample values are either send binary (default) or in ASCII format. In the first case, the target application has to have knowledge about the stream properties to correctly interpret the incoming byte stream. Due to the compact form it is the preferred choice when streaming at high sample rate. In ASCII mode sample values are converted in text and a line break is added after each sample.

```
<consumer create="ssi_consumer_SocketWriter" host="127.0.0.1" port="8888">
  <input pin="emph_t" frame="0.1s"/>
</consumer>
```

The according component for sending events is called SocketEventWriter. Again, the content of an event is either sent binary or converted into a text representation. However, in combination with another component introduced below, it can be used to share information from several streams and events in a single socket packet.

```
<object create="ssi_listener_SocketEventWriter" host="127.0.0.1" port="9999">
  <listen address="esr@xml"/>
</object>
```

Obviously, there is not *one* format which would suit any application we can think of. To allow a software to communicate with SSI either means that the application has to adapt to the format provided by SSI, or that SSI has to provide information in a format that is understood by the application. Of course, the latter is preferable as it reduces the effort for connecting the target application. To get as close as possible to this goal SSI provides a dynamic interface that can be adapted to fit a particular purpose. Once more, it is based on XML (see Section 5.5.3) which lends oneself as an ideal language due to its readability and prevalence. To make the most of the format and make it as flexible as possible, SSI does not prescribe a specific structure, but offers the possibility to define templates.

A template consists of a proper XML tree, which defines a static frame for wrapping up the desired information. At appropriate spots the XML tree is enriched with variables that will be filled with the corresponding information from the pipeline at run-time. Any stream or event data that is available to the component can be included. The variable $(stream=<x>), for example, will be filled with the input from the x'th stream. Adding the keyword select=<x,y,z,...> will restrict the output to a set of dimensions and setting mean=true will replace each dimension with the mean value of the stream. To include the content of a certain event, the variable $(event=<event>@<sender>) is available. By default the values that are linked with the event will be inserted and again select can be used to apply a selection. However, the keyword field also allows to pick other information, such as the time-stamp when the event has been created or the names that have been assigned to the values.

The following definition is just one possibility to format the output speech recogniser. All information is enclosed by <Emotion>, which includes the time-stamp as an attribute. Class probabilities are put in separate tags.

```
<Emotion time="$(event=emotion@voice;field=systemtime)">
  <NegativeActive>$(event=emotion@voice;select=0)</NegativeActive>
  <NegativePassive>$(event=emotion@voice;select=1)</NegativePassive>
  <Neutral>$(event=emotion@voice;select=2)</Neutral>
  <PositivePassive>$(event=emotion@voice;select=3)</PositivePassive>
  <PositiveActive>$(event=emotion@voice;select=4)</PositiveActive>
</Emotion>
```

The conversion is applied by XMLEventSender, which takes as option the template file and is set up to receive the events from the classifier. The result of the operation is a new event with address esr@xml. Since update is set to 0 an output is generated only if new input is received, otherwise an event is created every x milliseconds. Using SocketEventWriter the result can be shared with external applications.

```
<object create="ssi_consumer_XMLEventSender"
    ename="esr" sname="xml" path="template.xml" update="0">
  <listen address="emotion@voice"/>
</object>
```

The string that is received by a connected client now looks like:

```
<Emotion time="14600888">
    <NegativeActive>0.083193</NegativeActive>
    <NegativePassive>0.008975</NegativePassive>
    <Neutral>0.043374</Neutral>
    <PositivePassive>0.431801</PositivePassive>
    <PositiveActive>0.432657</PositiveActive>
</Emotion>
```

The complete code of the final system is available in the appendix (see A.2.4). A screenshot is shown in Figure 6.8.

6.6 Benefits

In the course of this chapter, we have assembled an emotional speech recogniser following the example of EMOVOICE [328]. But what are the benefits we have gained by converting EMOVOICE into an SSI pipeline?

As a classical command-line tool, EMOVOICE is configured via command-line parameters. By passing additional text when the program is launched we can change the behaviour of the application. Yet, possibilities are limited to the functions implemented in the pre-compiled executable.

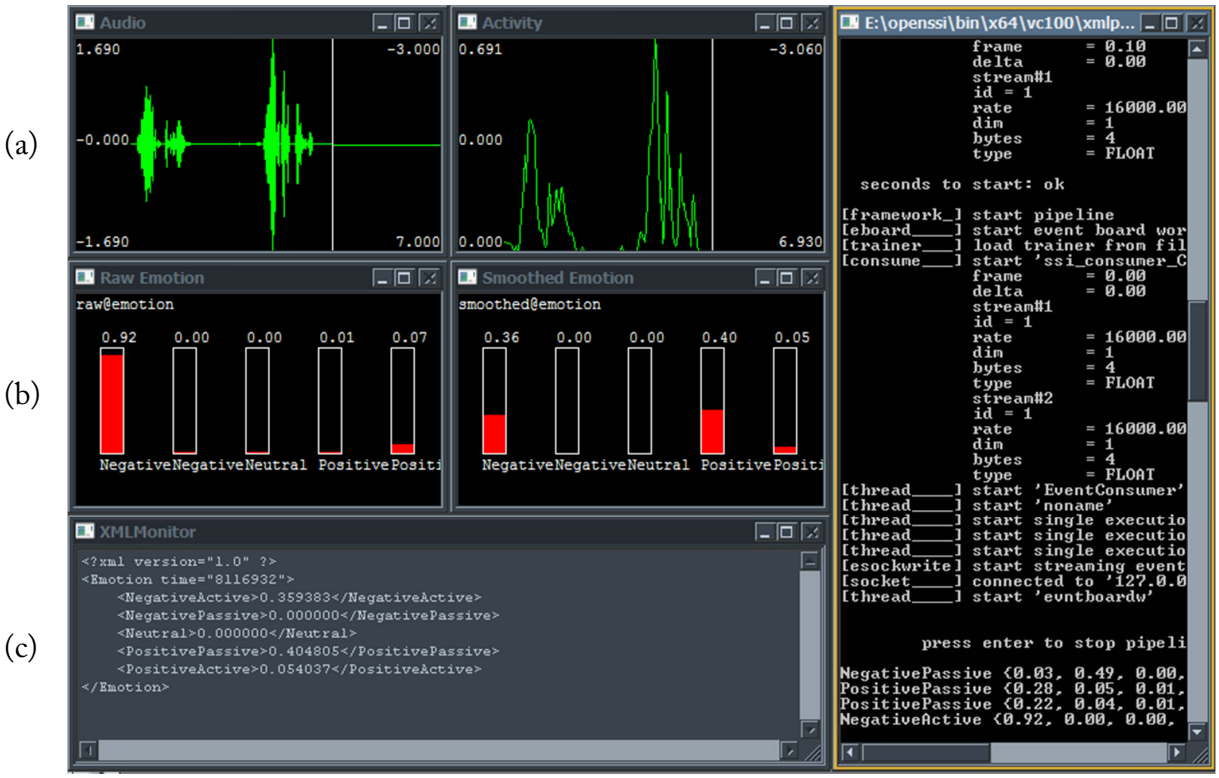


Figure 6.8: Screenshot of the final recognition system. It is composed of three main steps: (a) voice activity detection is used to spot voiced parts in the audio signal and probabilities for each class are calculated; (b) likelihoods are interpolated over time to provide a continuous frame-by-frame reaction of the system; (c) interpolated results are collected in an xml tree and provided through a socket connection.

Needs beyond the original purpose require adaptation of the original source code, meaning to get oriented in hundreds of lines of code, finding appropriate spots and applying necessary changes. As a consequence, it is not uncommon that developers decide to start a new project from scratch. And in fact, depending on how well the code is structured and documented, re-implementing can be faster than rewriting.

SSI tries to avoid this by introducing the concept of components and pipelines, which improves maintainability and reusability.

1. As introduced in Section 5.4.2 a component based architecture is followed to assemble pipelines fully modular. Maintaining single components is a lot easier than dealing with a project as a whole. And since components of the same type implement a common interface, they can be exchanged without touching the rest of the pipeline.

2. As introduced in Section 5.6.2 a plug-in system is featured, which allows loading components at run-time. This further decouples the system from its single components, since it makes it possible to inject a new version of a component without recompiling the pipeline.

3. Finally, as introduced in Section 5.6.7, pipelines can be represented as XML trees in which each component defines a node. This offers an additional level of abstraction and highly increases the transparency of the system, as it no longer comes as a black box controlled by a set of parameters, but is presented in a user-friendly, readable format that can be configured and edited in place.

During the previous example we have benefited from these concepts when we decided to use a second set of features or adding an additional interpolation step. To accomplish the same with the original EMOVOICE system, we had to go into the source code and apply the necessary changes there. Afterwards a rebuild of the whole project was required. Using SSI as backbone we only had to apply few minor changes to the XML structure of the pipeline[4]. In the same manner we could switch to another classification algorithm or even add input from additional modalities. In the next chapter we move on to a more complex detection system which combines asynchronous events from multiple modalities by applying an event-based fusion algorithm.

[4]Even if we have to write an according feature plugin first, we benefit from implementing an interface rather than touching external sources.

Chapter 7

Multimodal Enjoyment Recognition

In this chapter we describe a multimodal recognition system developed within the EU FET Project ILHAIRE[1] (Incorporating Laughter into Human Avatar Interactions: Research and Experiments). Although it is a significant feature of human communication, laughter is one of the least understood and most frequently human behaviours [266]. The objectives of ILHAIRE were to bridge the gap between knowledge on human laughter and its use in automatic emotion recognition and synthesis, thus enabling sociable conversational agents to understand and express natural-sounding laughter. In the course of the project SSI was employed to realise multiuser recording setups. In particular, we used the collected data to develop a novel event-driven fusion algorithm to recognise user enjoyment in real-time.

7.1 Belfast Storytelling Corpus

Laughter is a social phenomenon which occurs mainly in the presence of other individuals. And it is a behavioural-acoustic event which includes respiratory, vocal, and facial and skeletomuscular elements [266]. This makes data capturing a challenging task as it requires technologies to gather multimodal data from several individuals concurrently. The BELFAST STORYTELLING CORPUS [203], which was recorded using SSI as a backbone, is comprised of six sessions of groups of three or four people telling stories to one another in either English or Spanish. The storytelling task is based on the 16 Enjoyable Emotions Induction Task [145].

7.1.1 Setup

The participants of the study were provided with a description of Ekman's [96] 16 facets of enjoyable emotions and asked to think of a situation in their life, where they had experienced

[1] http://www.ilhaire.eu/

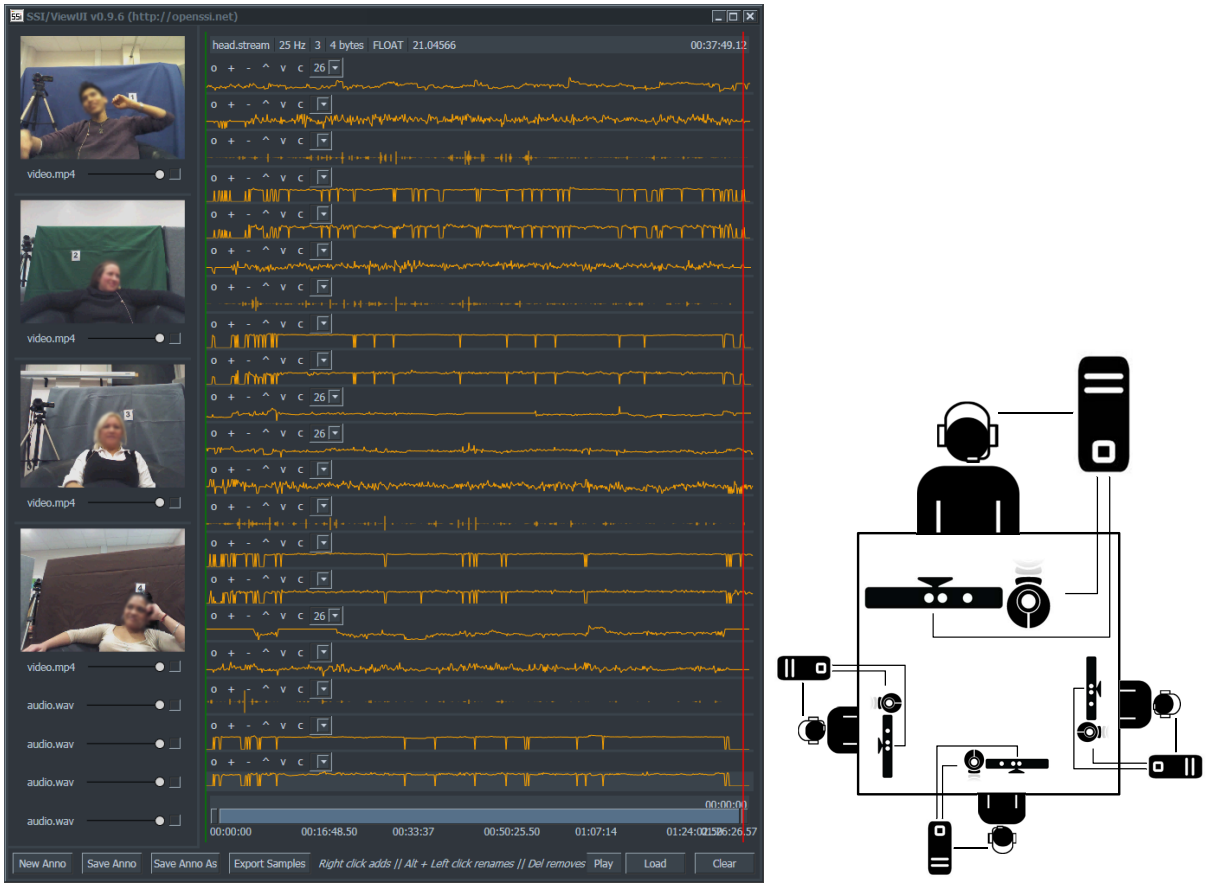


Figure 7.1: In the Belfast sessions four participants are recorded using Kinect, headset and webcam. Left: Signals collected in one session (faces in video blurred). Right: Setup sketch involving several computers synchronised via network broadcast.

each of the emotions. During the storytelling session the participants were seated in comfortable chairs around a central table. Participants took turns at recalling a story associated with each enjoyable emotion. The list of enjoyable emotions was randomised for each story telling session, and all of the participants told stories associated with the same emotion in each round of stories. The story-telling events occasionally evolved into an open discussion, which further aroused episodes of laughter.

During a recording session each participant wore a head-mounted microphone to capture high quality audio recordings. Video signals were recorded using Logitech Pro HD webcams. Kinect motion capture technology was used to capture facial features, gaze direction and depth information (see Figure 7.1). Webcam streams were compressed with the Huffyuv lossless codec and later compressed using the lossy H264 to make more usable file sizes. Since such a complex recording setup exceeds the capabilities of a standard PC, it was distributed over several machines using a host-client architecture in which multiple clients wait for a host to send a start command.

Sensor	#	Signal(s)	Sample Format	File Format
Kinect	4	Video	25 Hz, 640x480, RGB	avi
		Audio	16kHz, 32 bit	wav
		Skeleton	25 Hz, 240 float	ssi
		Depth Image	25 Hz, 640x480, 11 bit	ssi
		Facial action units	25 Hz, 6 float	ssi
		Facial points	25 Hz, 200 float	ssi
		Head movement	25 Hz, 3 float	ssi
Webcam	4	Video	25 Hz, 640x480, RGB	avi
Headset	4	Audio	48kHz, 24 bit	wav

Table 7.1: Listing of sensors and according signals in the Belfast setup.

7.1.2 Content

Each recording session lasted about 120 minutes, resulting in approximately 75 minutes recording time, and featured groups of three or four participants. Recorded signals are listed in Table 7.1. The amount of laughter varied depending on which emotion was being recalled and the nature of the story that was being recounted. The applied annotation scheme involves multiple tracks describing different laughter cues, such as voiced laughter or visual smiling. Isolated events are spanned by an additional track describing overall enjoyment, where enjoyment was defined enjoyment as an episode of positive emotion, indicated by visual and auditory cues of enjoyment, such as smiles and voiced laughters.

7.2 Event-driven Fusion

In Section 4.2 we have brought up the question whether for the task of affect recognition a segmented based combination of modalities can be regarded as an appropriate solution. The problem we see here is that forcing a decision over all involved modalities does not satisfy the complex temporal relations of social cues. Instead, we claim that an asynchronous combination of relevant events is needed (see Section 4.2.8). In the following we want to elaborate on this issue in more detail and present a novel event-driven fusion approach and compare it with standard fusion methods.

7.2.1 Cue Annotation

As a complex multimodal phenomenon laughter is well suited to study the impact of fusion. Its expression includes repetitive rhythmic shoulders and torso movements, visible inhalation, various facial actions and is often accompanied with some rhythmic as well as communicative

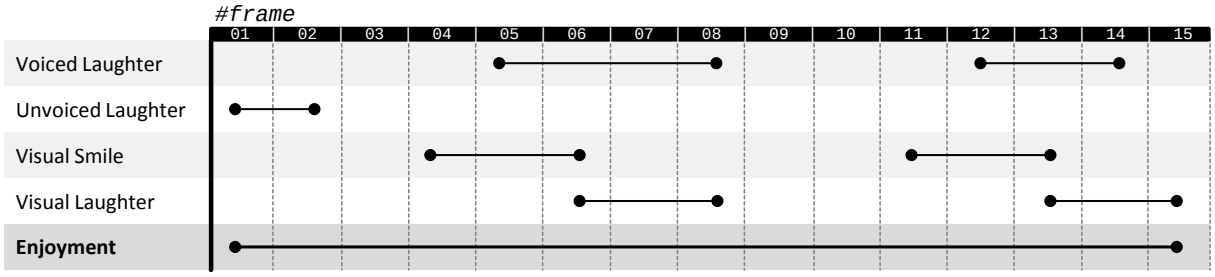


Figure 7.2: Exemplary annotation of a full enjoyment episode aligned with various voiced and visual cues emitted by the user. For each frame (bordered by dotted lines) a decision has to be made by the fusion system. In a conventional segmentation-based approach each frame is seen in isolation, i.e. a decision is derived from the multimodal information within the frame. However, we can see that the single cues only partly overlap with the enjoyment episode: While other frames align with cues from a single modality (see e.g. frame 2 and 4), some of the frames which are spanned by the enjoyment episode do actually not overlap with any observable cues (see e.g. frame 9 and 10). Those frames are likely to be misclassified by a segmentation-based approach. The event-driven fusion approach we propose here takes in account the temporal asynchronicity of the events, is able to overcome frames with sparse cues of enjoyment based on information of preceding frames.

gestures [227]. Through a complex temporal interplay these elements allow us to vary and control the quality and intensity of laughter. Instead of pure joy, for instance, there can be voluntary laughter, a blend of enjoyment with other emotions or attempts to suppress laughter [266]. In the following we use the term *enjoyment*, which we define as an episode of positive emotion, indicated by visual and auditory cues of enjoyment, such as smiles and voiced laughters. This enables us to derive a global measurement of enjoyment from accumulated indication-events.

The described experiment is based on the first session of the BELFAST STORYTELLING CORPUS. First, a finer annotation featuring detailed descriptions of enjoyment cues in all modalities is required. Figure 7.2 shows the various tracks introduced to cover visual and audible cues, such as smiles and laughs. The bottom track finally defines where an enjoyment episode starts and ends. The task of a classifier is to correctly predict enjoyment on a frame-to-frame basis. In classic segment-based fusion we would only take the annotation of enjoyment and train for each modality a classifier to output a probability whether for a particular frame the user is in an enjoyment state or not. The single outcomes are then combined to reach a final decision, for instance by summing them up. In contrast, the suggested event-driven approach uses the intermediate annotations to train classifiers that are specialised to certain cues. A final decision is then derived based on the cues that are currently present.

7.2.2 Motivation

Internally we represent the cues we detect as events, which we integrate asynchronously over time to form an overall decision. Introducing events as an abstract intermediate layer effectively decouples unimodal processing from the final decision making. Each modality serves as a client

which individually decides when to add information. Signal processing components can be added or replaced without having to touch the actual fusion system, and missing input from one of the modalities does not cause the collapse of the whole fusion process. In some sense this kind of event-driven fusion is similar to semantic fusion used to analyse the semantics of multimodal commands, and typically investigates the combination of gestures and speech in new-generation multimodal user interfaces [206].

The approach we adopt here is similar to that of Gilroy *et al.* in an artistic Augmented Reality installation called *Emotional Tree* [123]. They also use event-based fusion to derive the affective state of a user in real-time. The basic idea of their approach is to derive emotional information from different modality-specific sensors and map it onto a continuous affective space spanned by the three dimensions Pleasure, Arousal and Dominance (PAD model). Since the application depended on a continuous assessment of the affective user state, the current state of the fusion system was constantly represented by a vector in the PAD space. And the direction into which the vector would move was set by a bunch of vectors representing the single modality-specific contributions. The values of those guiding vectors was updated whenever a new affective cue was detected or otherwise decayed over time.

A different approach for predicting user affect in a continuous dimensional space based on verbal and non-verbal behavioural events (e.g. smiles, head shakes, or laughter), has been published by Eyben *et al.* [108]. In their system events are seen as "words", which are joined for each time segment and converted to a feature vector representation through a binary bag-of-words (BOW) approach. Tests on an audiovisual database proved the proposed string-based fusion to be superior over conventional feature-level modelling.

7.2.3 Algorithm

We only give a rough outline of the algorithm here, for details please see [198]. As already mentioned, the fusion scheme we propose is event-driven and bases on social cues, which are detected by independent classifiers. Detected cue events that enter the fusion process have to be mapped to a common representation. Here, we use a one dimensional space in which events are represented as vectors of a certain initial strength expressing the confidence that the user is in an enjoyment state (high value) or not (low value). Accordingly an event has either an increasing or decreasing effect on the current fusion output, which is also represented as a vector. This effect becomes less over time until at some point the vector is completely removed. If within short distance vectors of similar value pop up this leads to an reinforcing impact that bolsters the confidence of the fusion result, whereas contradictory cues will neutralise each other. If for some time no new events are detected the likelihood for the observed behaviour class automatically decreases and approaches a zero probability.

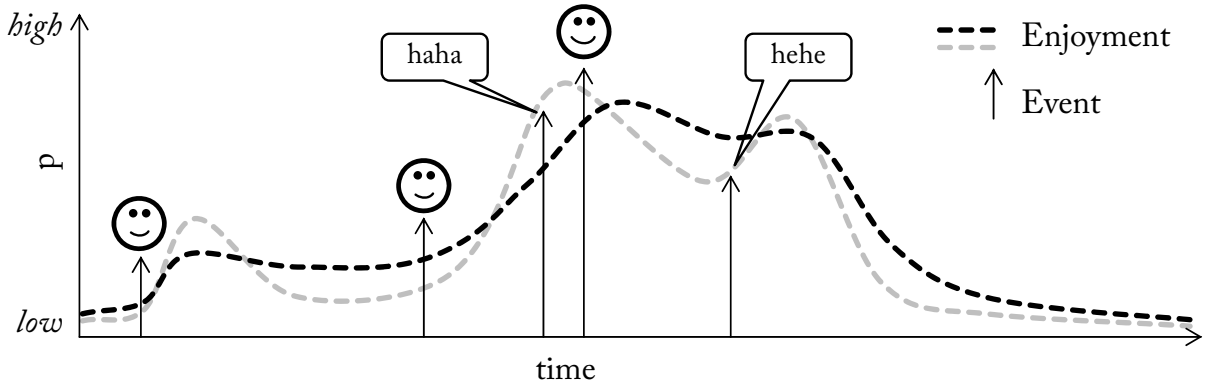


Figure 7.3: A fused score for enjoyment (dotted line) is derived from individual cues represented by vectors (vertical arrows) that express the confidence that the user is in a enjoyment state or not. Over time the strength of the vectors decreases and if no new cues are detected the likelihood for enjoyment approaches a zero probability. By adjusting weight and speed of the vectors it is possible to tune the fusion outcome. The picture illustrates the effect when the decay time of the cue vectors is decreased and at the same time the speed of the fusion vector is increased (gray line).

Three parameters are available to directly influence the performance of the fusion algorithm. The *vector weight* is a quantifier for the initial weighting the event has in the calculation of the fusion result. It is defined by the modality the event is detected in and serves as a regulation instrument for emphasising more reliable information sources. If, for example, one modality is generally better suited for the given classification problem, it can be assigned a higher overall weight. The weight can also be defined by the context. For example, in case of a high noise level, audio might be given less weight. The other parameters are the *decay speed* and the *fusion speed*. The first is also defined for each modality and describes the average lifespan of cues extracted from the respective signal. It determines the time it takes for the event's influence to decrease to zero and get discarded. Events that strongly indicate the fusion's target class can be given longer decay times, in order to prolong their influence on the result. The fusion speed, on the other, controls the inertial of the fusion vector. Increasing the speed leads to a faster reaction time, but may also increase jittering caused by false detections. Figure 7.3 illustrates the described vector fusion approach and the effect of parameter tuning.

7.3 Comparison

To compare segmentation based fusion with the described event-driven approach, we let all classification systems perform recognition on a framewise basis: A decision, if enjoyment is present within the evaluated person or not, is made every 400 milliseconds within a window of one second. We took approximately one hour recordings of two users for training (roughly 18.000 samples) and the recording of another user for testing (roughly) 9.000 samples. Considering

class imbalances within testing samples, we use the unweighted average as evaluation criterion, which is the average of the classwise recall values.

7.3.1 Tested Systems

Segmentation Based Fusion Segmentation based fusion approaches on the feature and decision level are applied to combine both modalities for direct enjoyment classification. From these experiments we can draw first conclusions if the multimodal information can deliver classification improvements, if the same annotated time segments are sliced through modalities.

Modality-Tailored Fusion Afterwards we try to recognise the annotated enjoyment segments indirectly from tailored annotations: instead of using the annotations for whole enjoyment episodes for both modalities, we annotate audible occurrences of laughters within the audio channel and visible laughters and smiles in the video separately. These tailored annotations are then used to train classification models for detecting these enjoyment indicating cues, rather than recognising enjoyment directly. Modality-tailored fusion is meant as an intermediate and experimental step, in which these modality tailored cue-recognisers are used directly in decision and model level fusion schemes. The models trained on enjoyment segmentations are therefore replaced, probabilities given to each frame by classifiers meant for detecting audible and visual laughters are mapped to the corresponding enjoyment classes.

Event Driven Vector Fusion Finally we apply event-based vector fusion to compare this indirect, event-based way of fusing multimodal information into single channel classification and segmentation based fusion performance. Before classification, activity recognition is performed for each modality, for example testing if there is more than noise in the audio channel. Such pre-processing can introduce additional prediction errors (as the activity recognition is also not always correct), but is a very crucial point in robust real-time systems. For this reason we simulate the process also for evaluation. Recognisers consider every frame that passes activity recognition and generates events for smiles and laughs respectively. Initial confidence values of these events correspond to the recognition probabilities of smile and laughter detectors. For every frame, we calculate the current position of the fusion vector and based on its current position we decide if enjoyment is present or not within the observed time frame.

7.3.2 Results

In order to simulate a true real-time system it should be noted that evaluation has been carried out for the full recordings, i.e. no frames were excluded at any time. Consequently, in case of

	Segmentation-Based		Tailored	Event-Driven
	Feature	*Decision*	*Decision*	*Vector Based*
Enjoyment	73.8%	76.4%	55.2%	76.2%
¬ Enjoyment	66.2%	61.5%	90.3%	81.4%
UA	**70.0%**	**69.0%**	**72.8%**	**78.8%**

Table 7.2: Results for segmentation-based fusion at feature and decision level (using overall enjoyment annotation) as well as modality tailored fusion (using intermediate annotation) and event-driven fusion (using vector fusion).

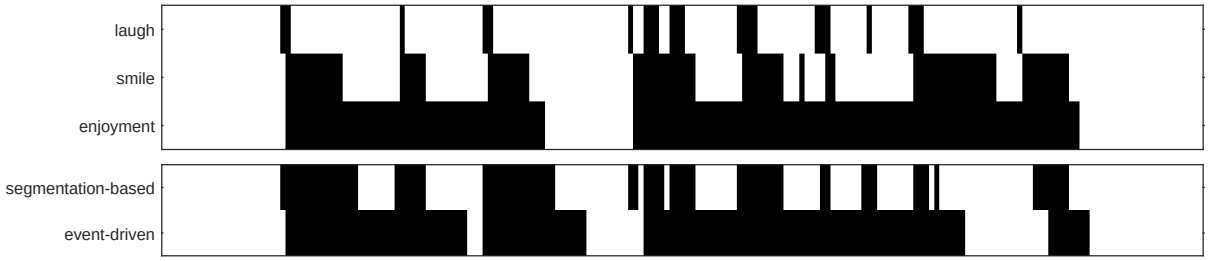


Figure 7.4: Comparison of annotation tracks and predicted labels (excerpt). Top: black rectangles mark enjoyment/cue annotations. Bottom: black rectangles mark enjoyment predictions. We see that the segmentation-based approach tends to miss enjoyment frames if there is neither a laughing nor a smiling cue. The event-driven approach, however, is able to partly bridge those gaps.

segmentation-based fusion a decision had to be forced even for frames where no signal was detected (i.e. no face tracked and silence detected in the audio channel). We decided to map those frames onto the class with the highest a priori probability (i.e. no enjoyment). As feature sets for characterising the raw audio streams, we use 1451 statistical prosodic EMOVOICE features [328]. Recognisers for video classification are trained with 36 features, gained from statistics over action units provided by the Microsoft Kinect SDK. As computational model for classification, we use LibSVM's support vector machines with a linear kernel [48].

Results are summarised in Table 7.2. When applying standard feature fusion 73.8% of the enjoyment frames are correctly predicted. In case of decision fusion (product rule) 76.4%. The UA is 70% and 69%, respectively. If overall enjoyment labels are replaced with tailored annotations, we observe a small improvement of $\sim$3%. However, the prediction of enjoyment frames drops to only 55.2%. In return, less than 10% of non enjoyment frames are missed. This makes perfect sense since the tailored models are tuned to detect cues of enjoyment rather than enjoyment episodes as a whole. Hence, the majority of enjoyment frames that do not overlap with a cue end up as false negatives. This once more reveals the weakness of segmentation-based fusion: forcing a decision over all modalities introduces noise if some of the modalities do not contain meaningful information, whereas the use of tailored annotations is suboptimal as they overlap only partly with the enjoyment episodes.

This unlocks potential for event-driven fusion. For one thing because information is only inte-

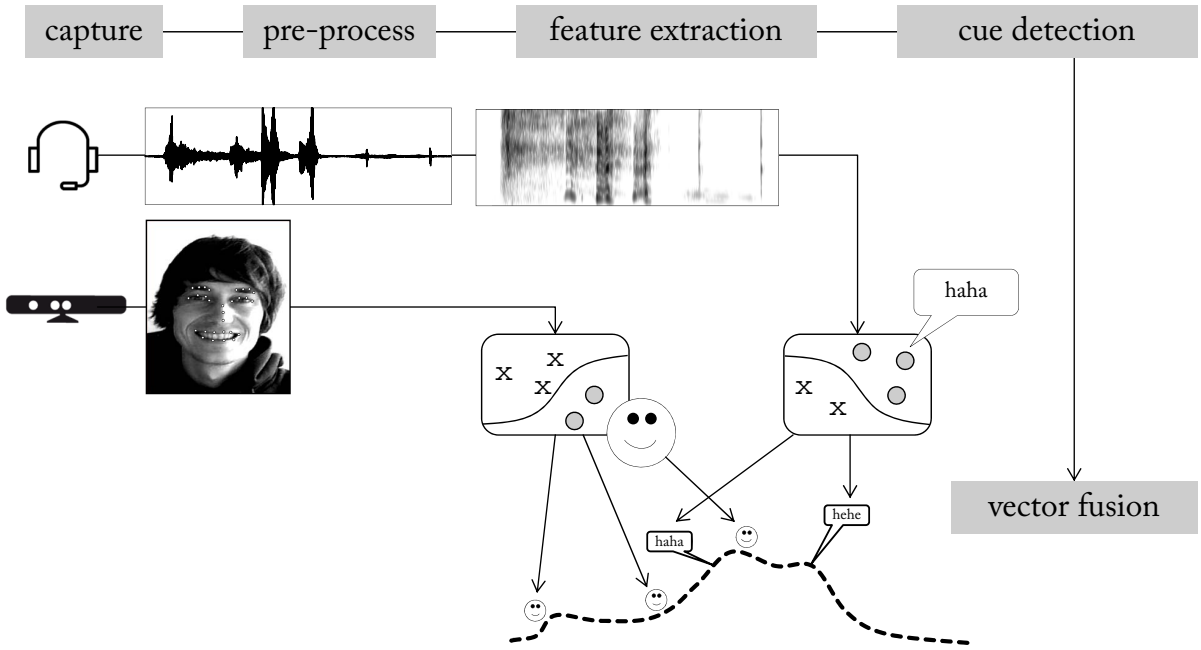


Figure 7.5: Schema of the multimodal enjoyment recognition system based on input from a microphone and the Kinect. Initially, both channels pass a classical machine learning pipeline to detect smiles and laughs. These cues are finally combined in a common vector space and translated to a continuous enjoyment level.

grated if it is regarded as relevant. For another thing because events do not directly affect the result. The outcome of the fusion process is rather an interpolation of the cues. This helps sorting out outliers and closing gaps between temporally close events. As a matter of fact, the vector-based approach yields an improvement of more than $\sim$8%. The assumption that this is mainly due to a better temporal modelling is confirmed in Figure 7.4. Here we see that the event-driven approach is able to correctly predict enjoyment even for frames not covered by an explicit laughing or smiling cue – at least as long as there is sufficient confidence in its surrounding. A more detailed discussion of the results is given in [198].

7.4 Multimodal Fusion System

In Section 6.5 we already have realised a fusion system with SSI. The system combined decisions stemming from two feature sets extracted from a single modality. Since feature extraction was applied over a common time window it represents a segmentation-based fusion approach. We will now modify the pipeline and turn it into a truly multimodal recognition system. Also, we will implement the event-driven fusion approach described in the last section. Figure 7.5 features a sketch of the intended multimodal system.

7.4.1 Second Modality

As a second sensor we add a Microsoft Kinect and access the stream with the action units (AUs) extracted from the user's face.

```
<sensor create="ssi_sensor_MicrosoftKinect" sr="25">
  <provider channel="au" pin="kinect_au"/>
</sensor>
```

Next, we add a transformer to convert the AUs into a compact feature set.

```
<transformer create="ssi_feature_MicrosoftKinectAUFeat">
  <input pin="kinect_au" frame="10" delta="15"/>
  <output pin="kinect_au_feat"/>
</transformer>
```

Finally, we add a classifier and connect it the smile model we have trained on the BELFAST STORYTELLING CORPUS.

```
<consumer create="ssi_consumer_Classifier" trainer="smile" ename="smile">
  <input pin="kinect_au_feat" frame="1"></input>
</consumer>
```

So far the structure resembles our earlier recognition system from Section 6.5, although it uses another feature set and model. We now combine the two systems and join them in a single pipeline. And we replace the audio model with the laughter model we have obtained in the last section. If we run the pipeline we could see smile and laugh events occurring independently of each other (but sharing a common time line).

7.4.2 Event Fusion

Before we can feed the events to the fusion process, we have to convert them into a suited format. Just like streams can be manipulated in series of connected transformers, SSI allows passing events from one component to another, possibly altering their content. Before an event enters the fusion process, we need to assign an initial weight in range 0 to 1. Since the SVM probabilities already meet this requirement. Nevertheless, we still need to select the correct dimension since the classifiers we use here output two probabilities – one that the cue has occurred and one for the complementary event.

```
<object create="ssi_listener_TupleSelect" sname="selector" indices="1">
  <listen address="laugh@classifier"/>
</object>
<object create="ssi_listener_TupleSelect" sname="selector" indices="0">
  <listen address="smile@classifier"/>
</object>
```

Finally, we configure the fusion component to listen to according events:

```
<object create="ssi_listener_VectorFusionModality"
  update_ms="400" fusionspeed="1.0f" path="fusion.modality"
  ename="enjoyment" sname="fusion">
  <listen address="laugh,smile@selector"/>
</object>
```

Several options allow us to adjust the speed of the fusion vector or the update rate, i.e. the frequency at which the fusion vector is recalculated. Individual speed and weight values are given in separate file (`fusion.modality`):

```
laugh@selector    0.06    0.7
smile@selector    0.1     0.9
```

The column contains the full event address, the second and third value define the speed and the weight. We can, for instance, see that smile events are given a higher speed than laughs. This is because laughs are a strong indicator of enjoyment and should therefore have a long-term influence on fusion; smiles on the other hand need quick reaction time as they describe well the margins of enjoyment. The exact values have been empirically determined, see [198] for a detailed discussion.

Figure 7.6 depicts a screenshot of the final system in action. The according pipeline is found in the Appendix A.3. A description of the full interactive system developed in ILHAIRE is given in [199].

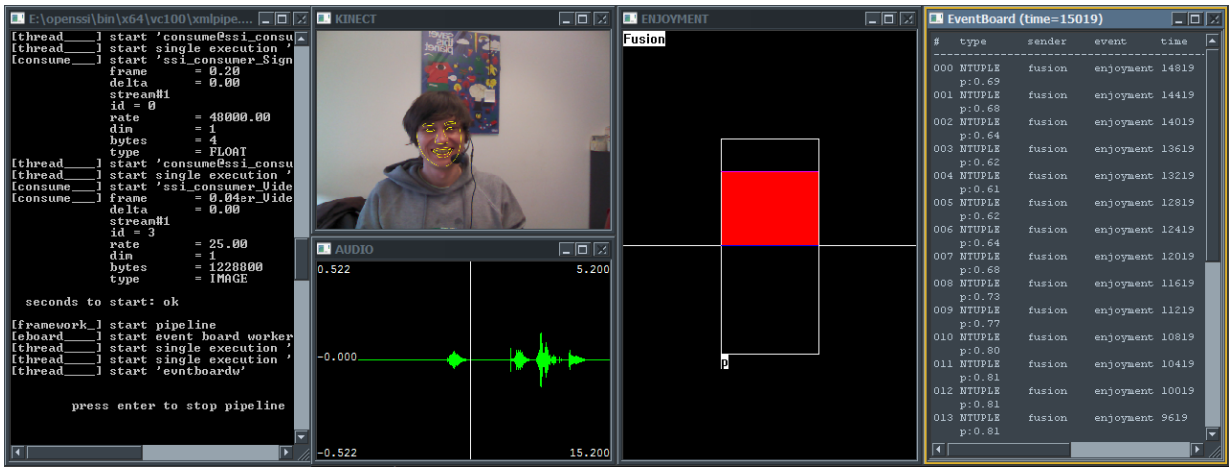


Figure 7.6: Multimodal enjoyment recognition.

Chapter 8

Conclusion

In recent years, *Affective Computing* [248] and *Social Signal Processing* (SSP) [242] have gradually emerged as new research fields in human-computer interaction. The growing interest is well reflected in the annual proceedings of *ACM Multimedia* (MM), one of the most influential conferences in the field of multimedia computing. In [7] Elisabeth André reviews the last 20 years starting with a special session on face and gesture recognition in 1998 followed by milestones like the first workshop on *Human-Centered Multimedia* in 2006 or the keynote by Pentland on *Honest Signals* in 2011. By 2012 more than one third of the *MM Grand Challenge* papers focused on topics related to emotions and in 2013 three workshops focusing on social and affective cues were held. It is not surprising that a conference like MM attracts more and more research to the field of SSP, since advances in this area have been highly dependent on the progress in video and audio processing. The detection of social cues from face, body and speech is key to build machines that are no longer perceived as "socially ignorant" but become sensitive to *nonverbal communication* [243, 321]. This does not only open up novel ways of interaction beyond conventional haptic-based control [239], but also leads to a new generation of "social computers" which will be perceived as more natural, efficacious and trustworthy [284, 324, 325].

The experiences of the last decade have shown that building such machines is an extremely challenging task. In order to fully exploit the potential of nonverbal communication, a system must be able to give prompt reactions to keep up the natural flow of interaction. This requires quick and efficient processing in (near) real-time. Moreover, the quality of the analysis should be independent of the environment in which the interaction takes place. In other words, a system should be able to detect social cues "in the wild", for instance while driving in a car or walking down a street, just as reliably as in front of a computer. Finally, it has to take into account that every human is an individual who is shaped by a certain cultural background, different personality traits, and his own personal values. Hence, despite a good deal of work that has been carried out in the area of affective computing and social signal processing, these efforts have not translated into many applications. Throughout the thesis, I have argued that this is not only due

to the complexity of the problem itself, but also due to a proliferation of studies that investigate social interaction under laboratory conditions. The problem we are facing is that the plain offline studies which still dominate literature convey too optimistic a picture of what can actually be achieved with a proposed system. This is because they avoid problems which occur only when a system is tested in the "open world".

8.1 Contributions

The goal of this thesis is to encourage developers to put more effort into building online systems instead of confining their work to pure offline studies. Thus we examined limitations of previous studies and proposed methodological and technical solutions which pave the path towards more realistic and application-oriented systems. As a major contribution, a novel open-source framework called SOCIAL SIGNAL INTERPRETATION (SSI) was introduced which supports the development of real-time applications in the area of SSP and affective computing.

1. **Constructing large multimodal corpora specifically aimed at the analysis of social phenomena.**

   Today, most SSP corpora are composed of discrete and exaggerated samples which do not reflect the variations of social behaviour in everyday life [80]. In Section 4.1 we have extensively discussed why a shift is needed towards material composed of natural expressions deliberately induced and presented in a continuous manner. Here, broadcast material, e.g. TV talk shows [322], Wizard-of-Oz scenarios [17] and game scenarios [147] have been identified as suitable sources. Since the richness of observable social expressions makes it impossible to obtain a single corpus for all social phenomena, data should be collected in as many different scenarios as possible including as many sensor devices as possible. To this end, a generic data structure is required which handles sensor streams independently of origin and content and goes beyond standard audiovisual recording setups and enrich them with additional measurements such as motion, eye gaze, and physiological feedback (Section 5.3.3). To give immediate access to the stored data sensor streams should be stored using common file formats such as AVI and CSV (Section 5.5). In order to keep captured signals in sync a mechanism has been proposed to synchronise signals without the need to time-stamp sample values or store them in an interleaved order (Section 5.4.3). The algorithm has been implemented in the SSI framework to synchronise data from a variety of sensor devices such as eye trackers, motion capture suites or physiological sensors possibly distributed over several machines (see Appendix A.4 for a listing of supported devices). SSI also offers the possibility to process captured streams on the fly and create supplementary descriptions to bolster following annotation steps (Section 6.3).

2. **Exploring novel ways to model multimodal fusion on multiple time scales and realize temporal correlations within and between different modalities.**

   In Section 4.2 we have argued that conventional fusion approaches, which have been developed on acted data, are not capable to properly model the complex temporal relationship between different modalities. In case of contradictory cues, for example, standard bimodal fusion approaches are as likely to decide in favour of a "correct" as an "incorrect" prediction [197]. The fact that large parts of naturalistic databases carry contradictory multimodal cues [81] explains that the gain in performance is generally lower for realistic scenarios [78]. Instead of forcing decisions over fixed time segments, *asynchronous* fusion approaches should be favoured to loosen the temporal constraints between modalities and allow them to individually decide when to contribute to the final decision making process. Several algorithms have been proposed to tackle this problem. Most of them are either variants of Hidden Markov Models (HMMs) [225, 304, 359] or are based on Artificial Neural Networks (ANNs) featuring some kind of memory cells [226, 348]. A disadvantage of those approaches is their complexity in terms of training and decision taking, in particular that there is no simple way to manually tune model parameters once the learning phase is finished. Event fusion defines an alternate approach by introducing events, for instance a raise in pitch or a visual smile, as an abstract intermediate layer to effectively decouple unimodal processing from the decision process. In Section 7.2 a fusion algorithm based on vector fusion has been proposed [198]. It makes the classification task more transparent and allows the developer to introduce knowledge into the process. A general problem of asynchronous fusion strategies, however, derives from the fact that – compared to conventional segmentation-based algorithms – they are more complex to implement and assess. Here, SSI provides a suitable testbed as it allows to fuse information at different levels of processing (early vs. late fusion) and provides facilities to implement and evaluate conventional segmentation based methods (Section 6.5) as well as complex asynchronous fusion algorithms (Section 7.2).

3. **Proposing a more application-related methodology to prepare the system for use in realistic environments.**

   As discussed in Section 4.3 it may require a good deal of work to turn an offline approach into an online application, which may still not perform satisfactory. The reason is that during the training phase the underlying data base is usually manually tweaked to suit the classification process, for instance by removing parts with sparse interaction and rare behaviour. An online system, however, has to deal with input as it comes. For instance, it must autonomously decide how to deal with non-prototypical behaviour [293]. Hence, ideally all recorded data should be considered during training and processed in a continuous frame-by-frame manner. The decision to skip one or more frames has to be made by the system based only on information that will be available at run-time, too. To this

end, SSI offers tools to simulate processing of pre-recorded data as if it was live input (see Section 6.4). Also, in offline studies the segmentation of the training data is often manually aligned to the observed behaviour. This, however, can lead to suboptimal results if the learning process does not fit the application. Hence, in order to translate well into the final system, the same segmentation algorithm should be applied during training as well. This was demonstrated in Section 6.3 when training data for a speech emotion recogniser was collected by steering textual instructions on the screen on-the-fly via voice activity detection.

4. **Introducing an open-source framework to accomplish multimodal real-time applications.**

   Eventually, it is not only the methodological differences which differ offline from online systems, but also a notable increase in implementation effort (Section 4.3.6). This is because signals can no longer be accessed as a whole, but only in form of small data chunks. This requires additional treatments, such as buffering of incoming signal values, parallel execution of serially connected processing units, and a proper synchronisation if multiple channels are involved. To encourage developers to shoulder this, overhead tools must be provided that take as much work off the hands of them as possible. For instance, a developer should only sketch how sensor input is processed while the framework takes care of the data flow. To support the complete process of the machine learning pipeline, it should be possible to handle data in form of continuous streams at a very low level (Section 5.3.3), but also high-level social behaviour in form of asynchronous events (Section 5.4.4). For a maximum impact, these tools should include the possibility of adding new functions (and a simple way to share them with other developers), but also offer a simple interface to users with little or no programming background. To this end, SSI provides a software architecture which accomplishes complex processing pipelines from simple, reusable units (Section 5.4). A plugin interface allows anybody to extend the pool of available components (Section 5.6) and an XML interface allows unskilled users to exploit the full potential of SSI (Section 5.6.7).

8.2 Applications

In recent years, SSI has been used by researchers around the world. The *EmoVoice* component [328] (as part of SSI) was used in various showcases in the European projects CALLAS[1] and IRIS[2] which analyses the expressivity in user speech. An example is the E-Tree [123], an Augmented Reality art installation of a virtual tree that grows, shrinks, changes colours, etc. by

[1] http://www.callas-newmedia.eu
[2] http://iris.interactive-storytelling.de

interpreting affective input from video, keywords and emotional voice tone. SSI was also employed to construct the CALLAS EXPRESSIVITY CORPUS [41, 43]. It consists of synchronised recordings from high-quality cameras and microphones, Nintendo's Wii remote controls and a data glove. Experiments have been conducted in Germany, Greece and Italy. In total, about 15 hours of interaction from more than 50 subjects was collected.

Within the AVLAUGHTERCYCLE project, which aimed at developing an audiovisual laughing machine, SSI was used for recording and laughter detection in real-time [317]. The work was continued within the EU-funded ILHAIRE[3] project. Here, SSI was used to collect the MMLI corpus [228] featuring 3D body position information, facial tracking, multiple audio and video channels of up to three interacting participants. In six sessions more than four hours of interaction was recorded involving 16 participants. In addition the BELFAST STORYTELLING CORPUS [203] was built, which is comprised of six sessions of groups of three or four people telling stories to one another in either English or Spanish. In total, it contains 25 hours and 40 minutes of high quality audio and video material (HD webcam and Microsoft's Kinect). Using the collected data, a multimodal enjoyment recognition system was built with SSI and embedded in an interactive game scenario [199, 318].

In TARDIS[4], another project under EU funding, a simulation platform for young people to improve their social skills was built. In this context, SSI was used for real-time detection of user's emotions and social attitudes through voice, body and facial expression recognition [6]. A visualisation tool called NOVA has been developed to visualise detected cues in graphs and heatmaps and give recommendations to the user [21, 22].

The EU funded CEEDS[5] project exploited the users' unconscious processes to optimise the presentation of large and complex databases. To this end, a sensor platform was developed on top of SSI [335] and integrated into the EXPERIENCE INDUCTION MACHINE (XIM), a multiuser mixed-reality space equipped with a number of sensors and effectors [25].

A research group at the Institute for Creative Technologies (ICT) from University of Southern California developed a multimodal sensor fusion framework on top of SSI called MULTISENSE, which they use to investigate the capabilities of automatic non-verbal behaviour descriptors to identify indicators of psychological disorders [278].

Outside an academic context, SSI was used at the Music Hack Day[6] 2013 in Barcelona to process physiological signals captured with the e-Health Sensor Platform for Arduino. During the hacking sessions participants were encouraged to use extracted user states to implement music related projects.

[3] http://www.ilhaire.eu/
[4] http://tardis.lip6.fr/
[5] http://ceeds-project.eu
[6] http://bcn.musichackday.org/2013

SSI is freely available under LGPL/GPL at `http://openssi.net`.

8.3 Future Work

Eventually, the success of online systems will depend on whether the idea is accepted by the user or not. A system that shows appropriate behaviour for most of the time but then fails at some crucial point in the interaction may be regarded as disturbing rather than helpful. In fact, there is the danger that strategies actually meant to ease the interaction for the user may cause the exact opposite. Going back to the example from the beginning, when we offered the old lady to call someone to pick her up, we cannot necessarily assume that she will actually like our proposal. A system that was asked for directions and decides to call an ambulance against the will of a person may not earn much sympathy.

In that sense, assessing the "real" quality of an online system is way more complicated than calculating the percentage of correct predictions for a set of preselected samples. In fact, even a correct prediction may be useless if it comes with delay and the right moment for a reaction has passed. On the other hand, a technically incorrect prediction might be tolerable if it triggers a behaviour that is still reasonable. Or a false prediction may remain without consequences if the system is able to recognise and correct its error. In the long run, an online system should be equipped with strategies to incorporate the user's feedback and should thus continuously asses the quality of the interaction. Otherwise it may not achieve its objective.

This, however, leads us to another crucial point. In classic machine learning applications, model parameters are fixed once the learning process is finished. For a system which is meant to interact with a user over a longer period this is hardly acceptable. In the first place, we cannot know if the current model parameters do actually fit a particular user (most likely they will not). And even if they do, the goals and expertise of the user may change over time and requires some kind of adaptation. An online system should therefore continuously adjust its behaviour to the user. This can be achieved by taking into account demographic information such as cultural background, gender or age, but also on the basis of previous interactions. This requires learning algorithms that permanently refine their model to adapt to the current state of the user.

Finally, the context of a current situation plays an essential role for the correct interpretation of social signals. A smile, for instance, could be taken as a sign of enjoyment, but may also express embarrassment or even contempt. Solving such ambiguities requires knowledge about the place an interaction takes place, the intensions and goals of the user, his current emotional state, his relation to potential interaction partners, and so on. Each of these variables can change the meaning and objective of a social interaction. From this point of view, it is fair to ask if "pure" machine learning algorithms will ever be able to learn such complex relations only from training samples or whether cognitively inspired models will be necessary to model them.

Although there are no definitive solutions for the raised questions, it is obvious that answers to them can only be found by putting systems “in the wild”. That is, building online systems and run them outside the lab in realistic environments for days or weeks. What we can learn from such an experiment may be worth a thousand offline studies.

Appendix A

Complete Code Examples

A.1 Basic Examples

A.1.1 Sensor

Header

```
#ifndef _MYSINESENSOR_H
#define _MYSINESENSOR_H

#include "base/ISensor.h"
#include "ioput/option/OptionList.h"
#include "thread/ClockThread.h"

namespace ssi {
// sinus wave generator
class MySineSensor : public ISensor, public ClockThread {
public:
  // options
  class Options : public OptionList {
  public:
    Options()
      : sr(50.0) { // set default value
      // define options by name, type and help string
      addOption("sr", &sr, 1, SSI_DOUBLE, "sample rate of sine wave");
    }
    // option variables to be used by the component
    double sr;
  };
  // channel class
  class MySineChannel : public IChannel {
```

```
  public:
    MySineChannel () {
      // define a 1-dimensional stream of float values by default sampled at 10 Hz
      ssi_stream_init(stream, 0, 1, sizeof(float), SSI_FLOAT, 10.0);
    }
    ~MySineChannel () {
      ssi_stream_destroy (stream);
    }
    // assign a name to the channel
    const ssi_char_t *getName() { return "sine"; };
    // access the stream
    const ssi_char_t *getInfo () { return "sine wave"; };
    ssi_stream_t getStream () { return stream; };
    // the stream
    ssi_stream_t stream;
  };
  // object interface
  MySineSensor(const ssi_char_t *file);
  ~MySineSensor();
  static const ssi_char_t *GetCreateName () { return "ssi_sensor_MySineSensor"; };
  static IObject *Create (const ssi_char_t *file) { return new MySineSensor (file); };
  Options *getOptions () { return &_options; };
  const ssi_char_t *getName () { return GetCreateName (); };
  const ssi_char_t *getInfo () { return "sine wave generator"; };
  // returns the number of channels
  ssi_size_t getChannelSize () { return 1; };
  // returns a channel
  IChannel *getChannel (ssi_size_t index) { return &_channel; };
  // sets the provider for a channel
  bool setProvider (const ssi_char_t *name, IProvider *provider);
  // connect, start, stop and disconnect sensor
  bool connect ();
  bool start ();
  bool stop ();
  void clock ();
  bool disconnect ();

protected:
  // class variables
  ssi_char_t *_file;       // option file name
  Options _options;        // options
  MySineChannel _channel; // channel
  IProvider *_provider;    // provider
};
}
#endif
```

Source

```
#include "MySineSensor.h"

namespace ssi {

MySineSensor::MySineSensor(const ssi_char_t *file)
  : _provider(0),
  _file(0) {
  Thread::setName(getName());
  // if an option file is given, load options
  if (file) {
    if (!OptionList::LoadXML(file, _options)) {
      OptionList::SaveXML(file, _options);
    }
    _file = ssi_strcpy(file);
  }
}

MySineSensor::~MySineSensor() {
  // if an option file is given, save options
  if (_file) {
    OptionList::SaveXML(_file, _options);
    delete[] _file;
  }
}

bool MySineSensor::setProvider(const ssi_char_t *name, IProvider *provider) {
  // adjust sample rate from options
  _channel.stream.sr = _options.sr;
  // initialize channel
  provider->init(&_channel);
  // store pointer to provider
  _provider = provider;
  return true;
}

bool MySineSensor::connect() {
  // set thread clock
  setClockHz(_channel.stream.sr);
  return true;
}

bool MySineSensor::start() {
  // starts the thread
  return ClockThread::start();
};
```

```
void MySineSensor::clock() {
  // calculate next sample value
  float y = sin(2.0f*3.1416f*(ssi_time_ms()/1000.0f));
  // hand value over to provider
  _provider->provide(ssi_pcast(ssi_byte_t, &y), 1);
}

bool MySineSensor::stop() {
  // stops the thread
  return ClockThread::stop();
};

bool MySineSensor::disconnect() {
  return true;
}
}
```

A.1.2 Filter

Header

```
#ifndef _MYABSFILTER_H
#define _MYABSFILTER_H

#include "base/IFilter.h"

namespace ssi {
// converts stream to absolute values
class MyAbsFilter : public IFilter {
public:
  // object interface
  static const ssi_char_t *GetCreateName() { return "ssi_filter_MyAbsFilter"; };
  static IObject *Create(const ssi_char_t *file) { return new MyAbsFilter(); };
  IOptions *getOptions() { return 0; };
  const ssi_char_t *getName() { return GetCreateName(); };
  const ssi_char_t *getInfo() { return "applies absolute value"; };
  // called when new data becomes available
  void transform(ITransformer::info info,
    ssi_stream_t &stream_in,
    ssi_stream_t &stream_out,
    ssi_size_t xtra_stream_in_num = 0,
    ssi_stream_t xtra_stream_in[] = 0);
  // called to determine the number of dimensions in the output stream
  ssi_size_t getSampleDimensionOut(ssi_size_t sample_dimension_in);
  // called to determine the number of bytes per sample value in the output stream
```

```
  ssi_size_t getSampleBytesOut(ssi_size_t sample_bytes_in);
  // called to determine the type of the sample values in the output stream
  ssi_type_t getSampleTypeOut(ssi_type_t sample_type_in);
};

}
#endif
```

Source

```
#include "MyAbsFilter.h"

namespace ssi {

// determine the number of dimensions in the output stream
ssi_size_t MyAbsFilter::getSampleDimensionOut(ssi_size_t sample_dimension_in) {
  return sample_dimension_in;
}
// determine the number of bytes per sample value in the output stream
ssi_size_t MyAbsFilter::getSampleBytesOut(ssi_size_t sample_bytes_in) {
  return sample_bytes_in;
}
// determine the type of the sample values in the output stream
ssi_type_t MyAbsFilter::getSampleTypeOut(ssi_type_t sample_type_in) {
  return sample_type_in;
}
// called when new data becomes available
void MyAbsFilter::transform (ITransformer::info info,
  ssi_stream_t &stream_in,
  ssi_stream_t &stream_out,
  ssi_size_t xtra_stream_in_num,
  ssi_stream_t xtra_stream_in[]) {
  // pointer to first sample value
  float *src = ssi_pcast(float, stream_in.ptr);
  float *dst = ssi_pcast(float, stream_out.ptr);
  // iterate over samples and calculate absolute values
  for (ssi_size_t i = 0; i < stream_in.num * stream_in.dim; i++) {
    *dst++ = abs (*src++);
  }
}
}
```

A.1.3 Consumer

Header

```
#ifndef _MYPRINTCONSUMER_H
#define _MYPRINTCONSUMER_H

#include "base/IConsumer.h"
#include "ioput/file/File.h"

namespace ssi {
// prints stream on the console
class MyPrintConsumer : public IConsumer {
public:
  // object interface
  static const ssi_char_t *GetCreateName() { return "ssi_consumer_MyPrintConsumer"; };
  static IObject *Create(const ssi_char_t *file) { return new MyPrintConsumer(); };
  IOptions *getOptions() { return 0; };
  const ssi_char_t *getName() { return GetCreateName(); };
  const ssi_char_t *getInfo() { return "prints stream on console"; };
  // called when new data becomes available.
  void consume(IConsumer::info consume_info,
    ssi_size_t stream_in_num,
    ssi_stream_t stream_in[]);
};
}
#endif
```

Source

```
#include "MyPrintConsumer.h"

namespace ssi {

  void MyPrintConsumer::consume(IConsumer::info consume_info,
  ssi_size_t stream_in_num,
  ssi_stream_t stream_in[]) {
  // pointer to first sample value
  float *src = ssi_pcast(float, stream_in[0].ptr);
  // iterate over samples and print values on console
  for (ssi_size_t i = 0; i < stream_in[0].num; i++) {
    for (ssi_size_t j = 0; j < stream_in[0].dim; j++) {
      printf("%.2f ", *src++);
    }
    printf("\n");
  }
}
}
```

A.1.4 Pipeline

C++

```
#include "ssi.h"
#include "MySineSensor.h"
#include "MyAbsFilter.h"
#include "MyPrintConsumer.h"
using namespace ssi;

void main() {

  // register components
  Factory::RegisterDLL("../bin/ssiframe");
  Factory::RegisterDLL("../bin/ssigraphic");
  Factory::Register(MySineSensor::GetCreateName(), MySineSensor::Create);
  Factory::Register(MyAbsFilter::GetCreateName(), MyAbsFilter::Create);
  Factory::Register(MyPrintConsumer::GetCreateName(), MyPrintConsumer::Create);

  // get pointer to framework
  ITheFramework *frame = Factory::GetFramework();
  IThePainter *painter = Factory::GetPainter();

  // add a sensor instance
  MySineSensor *sensor = ssi_create(MySineSensor, 0, true);
  sensor->getOptions()->sr = 50.0;
  ITransformable *sine_t = frame->AddProvider(sensor, "sine");
  frame->AddSensor(sensor);

  // add a filter instance
  MyAbsFilter *filter = ssi_create(MyAbsFilter, 0, true);
  ITransformable *sine_abs_t = \
    frame->AddTransformer(sine_t, filter, "1");

  // add a consumer instance
  MyPrintConsumer *consumer = ssi_create(MyPrintConsumer, 0, true);
  frame->AddConsumer(sine_abs_t, consumer, "0.1s");

  // visualization
  SignalPainter *paint = ssi_create(SignalPainter, 0, true);
  paint->getOptions()->size = 5;
  paint->getOptions()->setName("SINE");
  frame->AddConsumer(sine_t, paint, "0.1s");
  paint = ssi_create(SignalPainter, 0, true);
  paint->getOptions()->size = 5;
  paint->getOptions()->setName("ABS(SINE)");
  frame->AddConsumer(sine_abs_t, paint, "0.1s");
```

```
  // run pipeline
  frame->Start();
  painter->MoveConsole(400, 0, 400, 300);
  painter->Arrange(1, 2, 0, 0, 400, 300);

  // wait for signal to stop pipeline
  frame->Wait();
  frame->Stop();

  // clean up
  frame->Clear();
  painter->Clear();
  Factory::Clear();
}
```

XML

```
<pipeline ssi-v="1">

  <!-- register components -->
  <register>
    <load name="ssimy"/>
    <load name="ssigraphic"/>
  </register>

  <!-- arrange windows -->
  <painter arrange="true" apos="1,2,0,0,400,300" console="true" cpos="
      400,0,400,300"/>

  <!-- sensor -->
  <sensor create="ssi_sensor_MySineSensor" sr="$(sine:sr)">
    <provider channel="sine" pin="sine_t"/>
  </sensor>

  <!-- filter -->
  <transformer create="ssi_filter_MyAbsFilter">
    <input pin="sine_t" frame="1"/>
    <output pin="sine_abs_t"/>
  </transformer>

  <!-- consumer -->
  <consumer create="ssi_consumer_MyPrintConsumer">
    <input pin="sine_abs_t" frame="0.1s"/>
  </consumer>

  <!-- visualization -->
```

```
  <consumer create="ssi_consumer_SignalPainter" size="5" name="SINE">
    <input pin="sine_t" frame="0.1s"/>
  </consumer>
  <consumer create="ssi_consumer_SignalPainter" size="5" name="ABS(SINE)">
    <input pin="sine_abs_t" frame="0.1s"/>
  </consumer>

</pipeline>
```

A.2 Emotional Speech Recognition

A.2.1 Basic

```
<pipeline ssi-v="1">

  <!-- register components -->
  <register>
    <load name="ssiaudio"/>
    <load name="ssiemovoice"/>
    <load name="ssigraphic"/>
    <load name="ssiioput"/>
    <load name="ssimodel"/>
  </register>

  <painter arrange="true" apos="1,1,0,0,600,400" console="true" cpos="
      600,0,300,400"/>

  <!-- sensor -->
  <sensor create="ssi_sensor_Audio" option="audio" sr="16000">
    <provider channel="audio" pin="audio_t"/>
  </sensor>

  <!-- filter -->
  <transformer create="ssi_filter_PreEmphasis" k="0.97">
    <input pin="audio_t" frame="0.01s"/>
    <output pin="emph_t"/>
  </transformer>

  <!-- feature -->
  <transformer create="ssi_feature_EmoVoiceFeat">
    <input pin="emph_t" frame="0.5s" delta="0.5s"/>
    <output pin="evfeat_t"/>
  </transformer>

  <!-- classifier -->
```

```
  <consumer create="ssi_consumer_Classifier" trainer="..\..\model\simple"
      console="true">
    <input pin="evfeat_t" frame="1"/>
  </consumer>

  <!-- visualization -->
  <consumer create="ssi_consumer_SignalPainter" size="10" type="2" name="
      Audio">
    <input pin="emph_t" frame="0.1s"/>
  </consumer>

</pipeline>
```

A.2.2 Event-based

```
<pipeline ssi-v="1">

  <!-- register components -->
  <register>
    <load name="ssiaudio"/>
    <load name="ssiemovoice"/>
    <load name="ssigraphic"/>
    <load name="ssiioput"/>
    <load name="ssimodel"/>
  </register>

  <painter arrange="true" apos="1,3,0,0,300,400" console="true" cpos="
      600,0,300,400"/>

  <!-- sensor -->
  <sensor create="ssi_sensor_Audio" option="audio" sr="16000">
    <provider channel="audio" pin="audio_t"/>
  </sensor>

  <!-- filter -->
  <transformer create="ssi_filter_PreEmphasis" k="0.97">
    <input pin="audio_t" frame="0.01s"/>
    <output pin="emph_t"/>
  </transformer>

  <!-- activity detection -->
  <transformer create="ssi_feature_AudioActivity" threshold="0.01">
    <input pin="emph_t" frame="0.03s" delta="0.015s"/>
    <output pin="activity_t"/>
  </transformer>
  <consumer create="ssi_consumer_ZeroEventSender" mindur="1.0" hangin="3"
      hangout="10" ename="active" sname="audio">
```

```
    <input pin="activity_t" frame="0.1s"/>
  </consumer>

  <!-- classifier -->
  <consumer create="ssi_consumer_Classifier" trainer="..\..\model\simple"
      ename="emotion" sname="voice" console="false">
    <input pin="emph_t" listen="active@audio">
      <transformer create="ssi_feature_EmoVoiceFeat"/>
    </input>
  </consumer>

  <!-- visualization -->
  <consumer create="ssi_consumer_SignalPainter" size="10" type="2" name="
      Audio">
    <input pin="emph_t" frame="0.1s"/>
  </consumer>
  <consumer create="ssi_consumer_SignalPainter" size="10" type="0" name="
      Activity">
    <input pin="activity_t" frame="0.1s"/>
  </consumer>
  <consumer create="ssi_consumer_SignalPainter" size="0" type="2" name="
      Trigger">
    <input pin="emph_t" listen="active@audio"/>
  </consumer>

  <!-- monitor -->
  <object create="ssi_listener_EventMonitor" mpos="300,0,300,400">
    <listen address="emotion@voice" span="10000"/>
  </object>

</pipeline>
```

A.2.3 Recording

```
<?xml version="1.0" encoding="utf-16" standalone="yes"?>
<pipeline>

  <register>
    <load name="ssiaudio" />
    <load name="ssibrowser" />
    <load name="ssigraphic"/>
    <load name="ssiioput"/>
  </register>

  <!-- set framework options -->
  <framework console="true" cpos="0,0,400,600"/>
```

```
  <painter arrange="true" apos="1,1,0,0,400,300" console="true" cpos="
      0,300,400,300"/>

  <!-- sensor -->
  <sensor create="ssi_sensor_Audio" option="audio" sr="16000">
    <provider channel="audio" pin="audio_t"/>
  </sensor>

  <!-- filter -->
  <transformer create="ssi_filter_PreEmphasis" k="0.97">
    <input pin="audio_t" frame="0.01s"/>
    <output pin="emph_t"/>
  </transformer>

  <!-- activity detection -->
  <transformer create="ssi_feature_AudioActivity" threshold="0.01">
    <input pin="emph_t" frame="0.03s" delta="0.015s"/>
    <output pin="activity_t"/>
  </transformer>
  <consumer create="ssi_consumer_ZeroEventSender" sname="audio" ename="
      active" mindur="1.0" hangin="3" hangout="10" empty="false">
    <input pin="activity_t" frame="0.1s"/>
  </consumer>

  <!-- stimuli -->
  <object create="ssi_object_Stimuli" sname="stimuli" ename="url"
    folder="..\..\stimuli\velten"
    annoPath="..\..\data\esr_$(date)">
    <listen address="active@audio"/>
  </object>
  <object create="ssi_object_Browser" pos="400,0,400,600">
    <listen address="url@stimuli"/>
  </object>

  <!-- storage -->
  <consumer create="ssi_consumer_WavWriter" path="..\..\data\esr_$(date)">
    <input pin="emph_t" frame="0.1s"/>
  </consumer>

  <!-- visualization -->
  <consumer create="ssi_consumer_SignalPainter" size="10" type="2" name="
      Audio">
    <input pin="emph_t" frame="0.1s"/>
  </consumer>

</pipeline>
```

A.2.4 Advanced

```
<pipeline ssi-v="1">

  <!-- register components -->
  <register>
    <load name="ssiaudio"/>
    <load name="ssiemovoice"/>
    <load name="ssipraat"/>
    <load name="ssisignal"/>
    <load name="ssifusion"/>
    <load name="ssigraphic"/>
    <load name="ssiioput"/>
    <load name="ssimodel"/>
  </register>

  <painter arrange="true" apos="2,2,0,0,600,400" console="true" cpos="
      600,0,300,600"/>

  <!-- sensor -->
  <sensor create="ssi_sensor_Audio" option="audio" sr="16000">
    <provider channel="audio" pin="audio_t"/>
  </sensor>

  <!-- filter -->
  <transformer create="ssi_filter_PreEmphasis" k="0.97">
    <input pin="audio_t" frame="0.01s"/>
    <output pin="emph_t"/>
  </transformer>

  <!-- activity detection -->
  <transformer create="ssi_feature_AudioActivity" threshold="0.01">
    <input pin="emph_t" frame="0.03s" delta="0.015s"/>
    <output pin="activity_t"/>
  </transformer>
  <consumer create="ssi_consumer_ZeroEventSender" mindur="0.5" incdur="1.0"
      hangin="3" hangout="10" ename="active" sname="audio">
    <input pin="activity_t" frame="0.1s"/>
  </consumer>

  <!-- classifier -->
  <consumer create="ssi_consumer_Classifier" ename="raw" sname="emotion"
  trainer="..\..\model\ev+praat\esr">
  <input pin="emph_t" listen="active@audio"/>
  <xinput>
    <input pin="emph_t"/>
  </xinput>
```

```
</consumer>

<!-- smoother -->
<object create="ssi_object_DecisionSmoother" update="250" decay="0.02" speed
    ="0.1" ename="smoothed" sname="emotion">
  <listen address="raw@emotion"/>
</object>

<!-- visualization -->
<consumer create="ssi_consumer_SignalPainter" size="10" type="2" name="Audio
    ">
  <input pin="emph_t" frame="0.1s"/>
</consumer>
<consumer create="ssi_consumer_SignalPainter" size="10" type="0" name="
    Activity">
  <input pin="activity_t" frame="0.1s"/>
</consumer>
<object create="ssi_object_EventPainter" name="Raw␣Emotion" type="1"
    autoscale="false" reset="false" fix="1.0">
  <listen address="raw@emotion"/>
</object>
<object create="ssi_object_EventPainter" name="Smoothed␣Emotion" type="1"
    autoscale="false" reset="false" fix="1.0">
  <listen address="smoothed@emotion"/>
</object>

<!-- output -->
<consumer create="ssi_consumer_SocketWriter" host="127.0.0.1" port="8888">
  <input pin="emph_t" frame="0.1s"/>
</consumer>
<object create="ssi_consumer_XMLEventSender" ename="esr" sname="xml" path="
    template.xml" update="0" monitor="true" mpos="0,400,600,200">
  <listen address="smoothed@emotion"/>
</object>
<object create="ssi_listener_SocketEventWriter" host="127.0.0.1" port="9999"
    >
  <listen address="esr@xml"/>
</object>

</pipeline>
```

A.3 Multimodal Enjoyment Recognition

```
<pipeline ssi-v="1">

  <register>
```

```
    <load name="ssiaudio.dll"/>
    <load name="ssimicrosoftkinect.dll"/>
    <load name="ssisignal.dll"/>
    <load name="ssiemovoice.dll"/>
    <load name="ssimodel.dll"/>
    <load name="ssivectorfusion.dll"/>
    <load name="ssigraphic.dll"/>
  </register>

  <framework console="true" cpos="0,0,320,480"/>

  <!-- audio sensor and processing -->
  <sensor create="ssi_sensor_Audio" option="audio" sr="48000" scale="true">
    <provider channel="audio" pin="audio"/>
  </sensor>
  <transformer create="ssi_feature_EmoVoiceFeat" maj="2">
    <input pin="audio" frame="19200" delta="28800"/>
    <output pin="audio_feat"/>
  </transformer>
  <transformer create="ssi_feature_AudioActivity" threshold="0.025">
    <input pin="audio" frame="19200" delta="28800"/>
    <output pin="audio_activity"/>
  </transformer>

  <!-- kinect sensor and processing-->
  <sensor create="ssi_sensor_MicrosoftKinect" trackNearestPerson="true"
      showFaceTracking="true" showBodyTracking="false" sr="25">
    <provider channel="rgb" pin="kinect_rgb"/>
    <provider channel="au" pin="kinect_au"/>
    <provider channel="skeleton" pin="kinect_skel"/>
    <provider channel="face" pin="kinect_face"/>
  </sensor>
  <transformer create="ssi_feature_MicrosoftKinectAUFeat">
    <input pin="kinect_au" frame="10" delta="15"/>
    <output pin="kinect_au_feat"/>
  </transformer>
  <transformer create="ssi_feature_MicrosoftKinectFAD" minfaceframes="10">
    <input pin="kinect_face" frame="10" delta="15"/>
    <output pin="kinect_activity"/>
  </transformer>

  <!-- cue detection -->
  <consumer create="ssi_consumer_Classifier" trainer="smile" sname="
      classifier" ename="smile" console="false">
    <input pin="kinect_au_feat" frame="1" delta="0" trigger="kinect_activity
        "></input>
  </consumer>
```

```
  <consumer create="ssi_consumer_Classifier" trainer="laugh" ename="laugh"
      sname="audio" console="false">
    <input pin="audio_feat" frame="1" delta="0" trigger="audio_activity"></
        input>
  </consumer>

  <!-- event fusion -->
  <object create="ssi_listener_TupleSelect" ename="laugh" sname="selector"
      indices="1">
    <listen address="laugh@classifier"/>
  </object>
  <object create="ssi_listener_TupleSelect" ename="smile" sname="selector"
      indices="0">
    <listen address="smile@classifier"/>
  </object>
  <object create="ssi_listener_VectorFusionModality" dimension="1" update_ms
      ="400" fusionspeed="1.0f" eventspeed="0.0f" ename="enjoyment" sname="
      fusion" paint="true" paint_events="false" caption="p" wcaption="
      ENJOYMENT" path="fusion.modality" move="1,0,640,0,320,480">
    <listen address="laugh,smile@selector"/>
  </object>

  <!-- visualization -->
  <consumer create="ssi_consumer_SignalPainter" name="AUDIO" size="10" type=
      "2" move="1,0,320,240,320,240">
    <input pin="audio" frame="0.2s"/>
  </consumer>
  <consumer create="ssi_consumer_VideoPainter" flip="false" name="KINECT"
      move="1,0,320,0,320,240">
    <input pin="kinect_rgb" frame="1"/>
  </consumer>
  <object create="ssi_listener_EventMonitor" mpos="960,0,320,480" update="
      1000">
    <listen address="enjoyment@fusion" span="10000"/>
  </object>

</pipeline>
```

A.4 Sensor List

What to monitor	Channel	Sensor
Human sounds	Microphone	Audio input device Android smart phone / Tablet
Movement & Position of body (parts)	Accelerometer	Android smart phone / tablet Xsens MVN Biomech

		Wiimote Smartex Empatica WAX9 Alive heart monitor
	Gyroscope	Android smart phone / tablet WAX9
	Magnetometer	Android smart phone / tablet Xsens MVN Biomech Emotiv WAX9
Camera	Android smart phone / tablet	
		Video input device Kinect
	Depth sensor	Kinect Leap motion
	Barometric pressure sensor	
		Android smart phone / tablet WAX9
	mechanical pressure sensor Optical sensor mouse Capacitive sensor	SensingTex pressure mat Mouse Microsoft touch mouse (hand)
Body position (global)	GPS, Galileo, Glonass Cell tower positions Public Wifi positions Barometric pressure sensor	Android smart phone / tablet
Facial expressions	Camera	Android smartphone / tablet Video input device
	Depth sensor	Kinect
Eye movement / gaze	Eyetracker (camera)	The eye tribe (stationary) SMI Red (stationary) SMI Eye Tracking Glasses
	EOG sensor	Nexus-10 MkII
Heart rate (variability)	PPG sensor	Empatica
	BVP sensor	IOM e-Health board Nexus-10 MkII
	ECG sensor	Alive heart monitor e-Health board Nexus-10 MkII Smartex

Respiration	Chest / belly band	Nexus-10 MkII Smartex
	Airflow sensor	e-Health board
Skin conductance	Skin conductivity sensor	IOM e-Health board Nexus-10 MkII Empatica
Skin temperature	Temperature sensor	e-Health board Empatica WAX9
Blood oxygen saturation	SPO2 sensor	e-Health board
Brain activity	EEG sensor	Emotiv Nexus-10 MkII
Muscle activity	EMG sensor	Nexus-10 MkII e-Health board Myo armband

Bibliography

[1] E. Albornoz, M. Sánchez-Gutiérrez, F. Martinez-Licona, H. Rufiner, and J. Goddard, "Spoken emotion recognition using deep learning," in *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, ser. Lecture Notes in Computer Science, E. Bayro-Corrochano and E. Hancock, Eds. Springer International Publishing, 2014, vol. 8827, pp. 104–111.

[2] K. Albrecht, *Social Intelligence: The New Science of Success*. Wiley, 2006.

[3] J. Allwood, L. Cerrato, K. Jokinen, C. Navarretta, and P. Paggio, "The MUMIN coding scheme for the annotation of feedback, turn management and sequencing phenomena," *Language Resources and Evaluation*, vol. 41, no. 3-4, pp. 273–287, 2007.

[4] J. Alon, V. Athitsos, Q. Yuan, and S. Sclaroff, "A unified framework for gesture recognition and spatiotemporal gesture segmentation," *Pattern Analysis and Machine Intelligence*, vol. 31, no. 9, pp. 1685–1699, September 2009.

[5] N. Ambady and R. Rosenthal, "Thin slices of expressive behavior as predictors of interpersonal consequences: A meta-analysis," *Psychological Bulletin*, vol. 111, no. 2, pp. 256–274, 1992.

[6] K. Anderson, E. André, T. Baur, S. Bernardini, M. Chollet, E. Chryssafidou, I. Damian, C. Ennis, A. Egges, P. Gebhard, H. Jones, M. Ochs, C. Pelachaud, K. Porayska-Pomsta, P. Rizzo, and N. Sabouret, "The TARDIS framework: Intelligent virtual agents for social coaching in job interviews," in *International Conference of Advances in Computer Entertainment (ACE)*, 2013, pp. 476–491.

[7] E. André, "Exploiting unconscious user signals in multimodal human-computer interaction," *Multimedia Computing, Communications, and Applications*, vol. 9, no. 1s, pp. 48:1–48:5, October 2013.

[8] O. Aran and D. Gatica-Perez, "Fusing audio-visual nonverbal cues to detect dominant people in conversations," in *International Conference on Pattern Recognition (ICPR)*, August 2010.

[9] M. Argyle and M. Cook, *Gaze and mutual gaze*. Cambridge University Press, 1976.

[10] M. Argyle, V. Salter, H. Nicholson, M. Williams, and P. Burgess, "The communication of inferior and superior attitudes by verbal and non-verbal signals," *Scial and Clinical Psychology*, vol. 9, no. 3, pp. 222–231, 1970.

[11] M. Arnold, *Emotion and Personality*, no. 1. Columbia University Press, 1960.

[12] J. Aronoff, B. A. Woike, and L. M. Hyman, "Which are the stimuli in facial displays of anger and happiness? Configurational bases of emotion recognition," *Personality and Social Psychology*, vol. 62, no. 6, pp. 1050–1066, 1992.

[13] D. Arthur and S. Vassilvitskii, "K-Means++: the advantages of careful seeding," in *ACM-SIAM symposium on Discrete algorithms (SODA)*, Philadelphia, PA, USA, 2007, pp. 1027–1035.

[14] A. Atkinson, M. Tunstall, and W. Dittrich, "Evidence for distinct contributions of form and motion information to the recognition of emotions from body gestures," *Cognition*, vol. 104, no. 1, pp. 59–72, July 2007.

[15] J. R. Averill, *A Semantic Atlas of Emotional Concepts*. American Psychological Association, 1975.

[16] S. Basu, T. Choudhury, B. Clarkson, and A. Pentland, "Towards measuring human interactions in conversational settings," in *Workshop on Cues in Communication at Conference on Computer Vision and Pattern Recognition (CVPR)*, 2001.

[17] A. Batliner, K. Fischer, R. Huber, J. Spilker, and Nöth, "Desperately seeking emotions: Actors, wizards, and human beings," in *Workshop on Speech and Emotion: A Conceptual Framework for Research at International Symposium on Computer Architecture (ISCA)*, 2000.

[18] A. Batliner, K. Fischer, R. Huber, J. Spilker, and E. Nöth, "How to find trouble in communication," *Speech Communication*, vol. 40, pp. 117–143, 2003.

[19] A. Batliner, S. Steidl, D. Seppi, and B. Schuller, "Segmenting into adequate units for automatic recognition of emotion-related episodes: A speech-based approach," *Advances in Human-Computer Interaction*, vol. 2010, pp. 3:1–3:15, January 2010.

[20] A. Battocchi, F. Pianesi, and D. Goren-Bar, "DaFEx: Database of facial expressions." in *INTETAIN*, 2005, pp. 303–306.

[21] T. Baur, I. Damian, F. Lingenfelser, J. Wagner, and E. André, "NovA: Automated analysis of nonverbal signals in social interactions," in *Human Behavior Understanding*,

ser. Lecture Notes in Computer Science, A. Salah, H. Hung, O. Aran, and H. Gunes, Eds. Springer International Publishing, 2013, vol. 8212, pp. 160–171.

[22] T. Baur, G. Mehlmann, I. Damian, F. Lingenfelser, J. Wagner, B. Lugrin, E. André, and P. Gebhard, "Context-aware automated analysis and annotation of social human–agent interactions," *Interaction Intelligent System*, vol. 5, no. 2, pp. 11:1–11:33, June 2015.

[23] F. Beritelli, S. Casale, A. Russo, S. Serrano, and D. Ettorre, "Speech emotion recognition using MFCCs extracted from a mobile terminal based on ETSI front end," in *International Conference on Signal Processing*, November 2006.

[24] S. Bermejo and J. Cabestany, "Oriented principal component analysis for large margin classifiers," *Neural Networks*, vol. 14, no. 10, pp. 1447–1461, 2001.

[25] U. Bernardet, S. B. i Badia, A. Duff, M. Inderbitzin, S. L. Groux, W. A. Mansilla, Z. Mathews, A. Mura, A. Väljamäe, S. Wierenga, and P. F. M. J. Verschure, "The experience induction machine and its role in empirical research on presence," in *The 10th International Workshop on Presence*, 2008.

[26] D. Bernhardt, "Emotion inference from human body motion," University of Cambridge, Computer Laboratory, Technical Report UCAM-CL-TR-787, October 2010.

[27] L. Bessacier and J. Bonastre, "Frame pruning for speaker recognition," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 1998, pp. 765–768.

[28] S. Biersack and V. Kempe, "Exploring the influence of vocal emotion expression on communicative effectiveness," *Phonetica*, vol. 62, no. 2–4, pp. 106–119, 2005.

[29] C. M. Bishop, *Neural Networks for Pattern Recognition*, 1st ed. Oxford University Press, USA, January 1996.

[30] P. Blache, R. Bertrand, and G. Ferr, "Creating and exploiting multimodal annotated corpora: The ToMA project," in *Multimodal Corpora*, ser. Lecture Notes in Computer Science, M. Kipp, J.-C. Martin, P. Paggio, and D. Heylen, Eds. Springer Berlin Heidelberg, 2009, vol. 5509, pp. 38–53.

[31] R. A. Bolt, ""Put-that-there": Voice and gesture at the graphics interface," in *Conference on Computer Graphics and Interactive Techniques*, New York, NY, USA, 1980, pp. 262–270.

[32] W. Bosma and E. André, "Exploiting emotions to disambiguate dialogue acts," in *International Conference on Intelligent User Interfaces (IUI)*, New York, NY, USA, 2004, pp. 85–92.

[33] K. Bousmalis, M. Mehu, and M. Pantic, "Spotting agreement and disagreement: A survey of nonverbal audiovisual cues and tools," in *International Conference on Affective Computing and Intelligent Interaction and Workshops (ACII)*, Los Alamitos, September 2009.

[34] L. J. Brunner, "Smiles can be back channels," *Personality and Social Psychology*, vol. 37, no. 5, pp. 728–734, 1979.

[35] E. Brunswik, *Perception and the representative design of psychological experiments*. University of California Press, 1956.

[36] M. Buhrmester, T. Kwang, and S. D. Gosling, "Amazon's mechanical turk a new source of inexpensive, yet high-quality, data?" *Perspectives on psychological science*, vol. 6, no. 1, pp. 3–5, 2011.

[37] F. Burkhardt, A. Paeschke, M. Rolfes, W. Sendlmeier, and B. Weiss, "A database of german emotional speech," in *Conference of the International Speech Communication Association (INTERSPEECH)*, 2005, pp. 1517–1520.

[38] C. Busso and S. S. Narayanan, "Recording audio-visual emotional databases from actors: A closer look," in *International Conference on Language Resources and Evaluation (LREC)*, Marrakech, Morocco, May 2008, pp. 17–22.

[39] C. Busso, Z. Deng, S. Yildirim, M. Bulut, C. M. Lee, A. Kazemzadeh, S. Lee, U. Neumann, and S. Narayanan, "Analysis of emotion recognition using facial expressions, speech and multimodal information," in *International Conference on Multimodal Interfaces (ICMI)*, New York, NY, USA, 2004, pp. 205–211.

[40] C. Busso, M. Bulut, C.-C. Lee, A. Kazemzadeh, E. Mower, S. Kim, J. N. Chang, S. Lee, and S. Narayanan, "IEMOCAP: interactive emotional dyadic motion capture database." *Language Resources and Evaluation*, vol. 42, no. 4, pp. 335–359, 2008.

[41] G. Caridakis, J. Wagner, A. Raouzaiou, F. Lingenfelser, K. Karpouzis, and E. Andre, "A cross-cultural, multimodal, affective corpus for gesture expressivity analysis," *Multimodal User Interfaces*, vol. 7, no. 1-2, pp. 121–134, 2013.

[42] G. Caridakis, L. Malatesta, L. Kessous, N. Amir, A. Raouzaiou, and K. Karpouzis, "Modeling naturalistic affective states via facial and vocal expressions recognition," in *International Conference on Multimodal Interfaces (ICMI)*, New York, NY, USA, 2006, pp. 146–154.

[43] G. Caridakis, J. Wagner, A. Raouzaiou, Z. Curto, E. André, and K. Karpouzis, "A multimodal corpus for gesture expressivity analysis," in *International Conference on Language*

Resources and Evaluation (LREC), Multimodal Corpora: Advances in Capturing, Coding and Analyzing Multimodality, Malta, 5 2010.

[44] J. Carletta, S. Isard, G. Doherty-Sneddon, A. Isard, J. C. Kowtko, and A. H. Anderson, "The reliability of a dialogue structure coding scheme," *Computational Linguistics*, vol. 23, no. 1, pp. 13–31, March 1997.

[45] J. Carletta, S. Ashby, S. Bourban, M. Flynn, M. Guillemot, T. Hain, J. Kadlec, V. Karaiskos, W. Kraaij, M. Kronenthal, G. Lathoud, M. Lincoln, A. Lisowska, I. McCowan, W. Post, D. Reidsma, and P. Wellner, "The AMI meeting corpus: A pre-announcement," in *Machine Learning for Multimodal Interaction*, Berlin, Heidelberg, 2006, pp. 28–39.

[46] G. Castellano, L. Kessous, and G. Caridakis, "Emotion recognition through multiple modalities: Face, body gesture, speech." in *Affect and Emotion in Human-Computer Interaction*, ser. Lecture Notes in Computer Science, C. Peter and R. Beale, Eds. Springer, 2008, vol. 4868, pp. 92–103.

[47] F. Cavicchio and M. Poesio, "Multimodal corpora annotation: Validation methods to assess coding scheme reliability," in *Multimodal Corpora*, ser. Lecture Notes in Computer Science, M. Kipp, J.-C. Martin, P. Paggio, and D. Heylen, Eds. Springer Berlin Heidelberg, 2009, vol. 5509, pp. 109–121.

[48] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011, software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[49] T. L. Chartrand and J. A. Bargh, "The chameleon effect: the perception-behavior link and social interaction." *Personality and Social Psychology*, vol. 76, no. 6, pp. 893–910, 1999.

[50] C. Chelba, T. Brants, W. Neveitt, and P. Xu, "Study on interaction between entropy pruning and Kneser-Ney smoothing," in *Conference of the International Speech Communication Association (INTERSPEECH)*, 2010, pp. 2242–2245.

[51] L. Chen and T. Huang, "Emotional expressions in audiovisual human computer interaction," in *International Conference on Multimedia and Expo (ICME)*, 2000, pp. 423–426vol.1.

[52] L. Chen, T. Huang, T. Miyasato, and R. Nakatsu, "Multimodal human emotion/expression recognition," in *International Conference on Automatic Face and Gesture Recognition (FGR)*, 1998, pp. 366–371.

[53] C. Chiarcos, S. Dipper, M. Götze, U. Leser, A. Lüdeling, J. Ritz, and M. Stede, "A flexible framework for integrating annotations from different tools and tag sets," *Translation and Literature*, vol. 49, no. 2, pp. 217–246, 2008.

[54] H. Clark, "Using uh and um in spontaneous speaking," *Cognition*, vol. 84, no. 1, pp. 73–111, May 2002.

[55] M. G. Core and J. F. Allen, "Coding dialogs with the DAMSL annotation scheme," in *Working Notes of the AAAI Fall Symposium on Communicative Action in Humans and Machines*, Cambridge, MA, November 1997, pp. 28–35.

[56] M. Coulson, "Attributing emotion to static body postures: recognition accuracy, confusions, and viewpoint dependence," *Nonverbal Behavior*, vol. 28, no. 2, pp. 117–139, 2010.

[57] T. Cover and P. Hart, "Nearest neighbor pattern classification," *Information Theory*, vol. 13, no. 1, pp. 21–27, September 2006.

[58] R. Cowie, E. Douglas-Cowie, B. Apolloni, J. Taylor, A. Romano, and W. Fellenz, "What a neural net needs to know about emotion words," in *Computational Intelligence and Applications*, 1999, pp. 109–114.

[59] R. Cowie, E. Douglas-Cowie, N. Tsapatsoulis, G. Votsis, S. Kollias, W. Fellenz, and J. Taylor, "Emotion recognition in human-computer interaction," *IEEE Signal Processing Magazine*, vol. 18, no. 1, pp. 32–80, 2001.

[60] R. Cowie, E. Douglas-Cowie, and C. Cox, "Beyond emotion archetypes: Databases for emotion modelling using neural networks," *Neural Networks*, vol. 18, no. 4, pp. 371+, 2005.

[61] R. Cowie and R. R. Cornelius, "Describing the emotional states that are expressed in speech," *Speech Communication*, vol. 40, no. 1-2, pp. 5–32, April 2003.

[62] E. Cox, "Adaptive fuzzy systems," *Spectrum*, vol. 30, no. 2, pp. 27–31, 1993.

[63] E. Crane and M. Gross, "Motion capture and emotion: Affect detection in whole body movement," in *Affective Computing and Intelligent Interaction*, ser. Lecture Notes in Computer Science, A. Paiva, R. Prada, and R. Picard, Eds. Springer Berlin / Heidelberg, 2007, vol. 4738, pp. 95–101.

[64] N. Crook, D. Field, C. Smith, S. Harding, S. Pulman, M. Cavazza, D. Charlton, R. Moore, and J. Boye, "Generating context-sensitive ECA responses to user barge-in interruptions," *Multimodal User Interfaces*, vol. 6, no. 1-2, pp. 13–25, 2012.

[65] N. Dael, M. Mortillaro, and K. Scherer, "The body action and posture coding system (BAP): Development and reliability," *Nonverbal Behavior*, vol. 36, no. 2, pp. 97–121, 2012.

[66] A. Daems and K. Verfaillie, "Viewpoint-dependent priming effects in the perception of human actions and body postures," *Visual Cognition*, vol. 6, no. 6, pp. 665–693, December 1999.

[67] G. E. Dahl, J. W. Stokes, L. Deng, and D. Yu, "Large-scale malware classification using random projections and neural networks." in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2013, pp. 3422–3426.

[68] A. R. Damasio, *Looking for Spinoza: joy, sorrow, and the feeling brain*, vol. 1st. Harcourt, 2003.

[69] B. Dasarathy, "Sensor fusion potential exploitation-innovative architectures and illustrative applications," *Proceedings of the IEEE*, vol. 85, no. 1, pp. 24–38, Jan 1997.

[70] S. Davis and P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," *Acoustics Speech and Signal Processing*, vol. 28, no. 4, pp. 357–366, 1980.

[71] L. De Silva and P. C. Ng, "Bimodal emotion recognition," in *International Conference on Automatic Face and Gesture Recognition (FGR)*, 2000, pp. 332–335.

[72] L. De Silva, T. Miyasato, and R. Nakatsu, "Facial emotion recognition using multi-modal information," in *International Conference on Information, Communications and Signal Processing*, September 1997, pp. 397–401vol.1.

[73] L. C. De Silva, L.c., T. Miyasato, and R. Nakatsu, "Use of multimodal information in facial emotion recognition," *Information and Systems*, vol. 81, no. 1, pp. 105–114, January 1998.

[74] C. Demiroglu, D. V. Anderson, and M. A. Clements, "A missing data-based feature fusion strategy for noise-robust automatic speech recognition using noisy sensors." in *International Symposium on Circuits and Systems (ISCAS)*, 2007, pp. 965–968.

[75] J. Deng, Z. Zhang, and B. W. Schuller, "Linked source and target domain subspace feature transfer learning - exemplified by speech emotion recognition," in *22nd International Conference on Pattern Recognition, ICPR 2014, Stockholm, Sweden, August 24-28, 2014*, 2014, pp. 761–766.

[76] L. Deng and D. Yu, "Deep learning: Methods and applications," *Foundations and Trends in Signal Processing*, vol. 7, no. 3, pp. 197–387, June 2014.

[77] K. Dion, E. Berscheid, and E. Walster, "What is beautiful is good." *Personality and Social Psychology*, vol. 24, no. 3, pp. 285–290, 1972.

[78] S. D'Mello and J. Kory, "Consistent but modest: A meta-analysis on unimodal and multimodal affect detection accuracies from 30 studies," in *International Conference on Multimodal Interaction (ICMI)*, New York, NY, USA, 2012, pp. 31–38.

[79] E. Douglas-Cowie, R. Cowie, and M. Schröder, "A new emotion database: Considerations, sources and scope," in *ISCA Workshop on Speech and Emotion: A Conceptual Framework for Research*, Belfast, 2000, pp. 39–44.

[80] E. Douglas-Cowie, N. Campbell, R. Cowie, and P. Roach, "Emotional speech: Towards a new generation of databases," *Speech Communication*, vol. 40, no. c, pp. 33–60, 2003.

[81] E. Douglas-Cowie, L. Devillers, J.-C. Martin, R. Cowie, S. Savvidou, S. Abrilian, and C. Cox, "Multimodal databases of everyday emotion: facing up to complexity." in *Conference of the International Speech Communication Association (INTERSPEECH)*, 2005, pp. 813–816.

[82] E. Douglas-Cowie, R. Cowie, I. Sneddon, C. Cox, O. Lowry, M. McRorie, J.-C. Martin, L. Devillers, S. Abrilian, A. Batliner, N. Amir, and K. Karpouzis, "The HUMAINE database: Addressing the collection and annotation of naturalistic and induced emotional data." in *International Conference on Affective Computing and Intelligent Interaction (ACII)*, 2007, pp. 488–500.

[83] E. Douglas-Cowie, R. Cowie, C. Cox, N. Amier, and D. Heylen, "The sensitive artificial listner: an induction technique for generating emotionally coloured conversation," in *LREC Workshop on Corpora for Research on Emotion and Affect*, Paris, France, 2008, pp. 1–4.

[84] H. Dreyfus and S. Dreyfus, "What artificial experts can and cannot do," *AI & SOCIETY*, vol. 6, no. 1, pp. 18–26, 1992.

[85] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*. New Yotk: John Willey & Sons, 1973.

[86] P. Duhamel and M. Vetterli, "Fast Fourier transforms: A tutorial review and a state of the art," *Signal Process.*, vol. 19, no. 4, pp. 259–299, April 1990.

[87] M. Dunnachie, P. Shields, D. Crawford, and M. Davies, "Filler models for automatic speech recognition created from hidden Markov models using the K-Means algorithm," in *European Signal Processing Conference (EUSIPCO)*, August 2009, pp. 544–548.

[88] N. Eagle and A. Pentland, "Reality mining: Sensing complex social systems," *Personal and Ubiquitous Computing*, vol. 10, no. 4, pp. 255–268, March 2006.

[89] A. Eerekoviae, "An insight into multimodal databases for social signal processing: acquisition, efforts, and directions," *Artificial Intelligence Review*, vol. 42, no. 4, pp. 663–692, 2014.

[90] P. Ekman, *Emotions revealed: recognizing faces and feelings to improve communication and emotional life*, ser. A Holt paperback. Henry Holt, 2007.

[91] P. Ekman, *The Philosophy of Deception: Lie Catching and Micro Expressions*. Ed. Clancy Martin, Oxford University Press, 2009.

[92] P. Ekman and W. Friesen, *Facial Action Coding System: A Technique for the Measurement of Facial Movement*. Palo Alto: Consulting Psychologists Press, 1978.

[93] P. Ekman and W. V. Friesen, "Nonverbal leakage and clues to deception," *Psychiatry*, vol. 32, no. 1, pp. 88–106, February 1969.

[94] P. Ekman, R. J. Davidson, and W. V. Friesen, "The duchenne smile: emotional expression and brain physiology," *Personality and Social Psychology*, vol. 58, no. 2, pp. 342–353, 1990.

[95] P. Ekman, "An argument for basic emotions," *Cognition & Emotion*, vol. 6, no. 3, pp. 169–200, 1992.

[96] P. Ekman, *Emotions revealed: recognizing faces and feelings to improve communication and emotional life*. New York: Times Books, 2003.

[97] P. Ekman and W. V. Friesen, "The repertoire of nonverbal behavior: Categories, origins, usage, and coding," *Semiotica*, vol. 1, pp. 49–98, 1969.

[98] P. Ekman and W. V. Friesen, "Constants across cultures in the face and emotion," *Personality and Social Psychology*, vol. 17, no. 2, pp. 124–129, 1971.

[99] P. Ekman and W. V. Friesen, *Unmasking the face: A guide to recognizing emotions from facial clues*. Oxford: Prentice-Hall, 1975.

[100] R. el Kaliouby and P. Robinson, "Generalization of a vision-based computational model of mind-reading," in *International Conference on Affective Computing and Intelligent Interaction (ACII)*, Berlin, Heidelberg, 2005, pp. 582–589.

[101] I. S. Engberg, A. V. Hansen, O. Andersen, and P. Dalsgaard, "Design, recording and verification of a danish emotional speech database," in *European Conference on Speech Communication and Technology (EUROSPEECH)*, 1997.

[102] I. Engleberg and D. Wynn, *Working in Groups: Communication Principles and Strategies*, ser. My Communication Kit Series. Allyn & Bacon, Incorporated, 2006.

[103] D. Erickson, O. Fujimura, and B. Pardo, "Articulatory correlates of prosodic control: Emotion and emphasis," *Language and Speech*, vol. 41, no. 3–4, pp. 399–417, 1998.

[104] D. Erickson, C. Menezes, and A. Fujino, "Some articulatory measurements of real sadness." in *Conference of the International Speech Communication Association (INTERSPEECH)*, 2004.

[105] I. Essa and A. Pentland, "Coding, analysis, interpretation, and recognition of facial expressions," *Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 757–763, 1997.

[106] R. Exline, *Explorations in the process of person perception: visual interaction in relation to competition, sex, and need for affiliation*. Defense Technical Information Center, 1963.

[107] F. Eyben, M. Wöllmer, and B. W. Schuller, "OpenSMILE: the munich versatile and fast open-source audio feature extractor," in *International Conference on Multimedia (MM)*, 2010, pp. 1459–1462.

[108] F. Eyben, M. Wöllmer, M. F. Valstar, H. Gunes, B. Schuller, and M. Pantic, "String-based audiovisual fusion of behavioural events for the assessment of dimensional affect," in *International Conference on Automatic Face and Gesture Recognition (FGR)*, USA, March 2011, pp. 322–329.

[109] F. Eyben, F. Weninger, F. Gross, and B. Schuller, "Recent developments in OpenSMILE, the munich open-source multimedia feature extractor," in *International Conference on Multimedia (MM)*, New York, NY, USA, 2013, pp. 835–838.

[110] S. Fagel, "Emotional McGurk effect," in *International Conference on Speech Prosody*, Dresden, Germany, May 2006.

[111] G. Fant, "Some problems in voice source analysis," *Speech Communication*, vol. 13, pp. 7–22, 1993.

[112] M. Farrús, J. Hernando, and P. Ejarque, "Jitter and shimmer measurements for speaker recognition," in *Conference of the International Speech Communication Association (INTERSPEECH)*, 2007, pp. 778–781.

[113] J.-M. Fernandez-Dols and M.-A. Ruiz-Belda, "Are smiles a sign of happiness? Gold medal winners at the olympic games," *Personality and Social Psychology*, vol. 69, no. 6, pp. 1113–1119, 1995.

[114] P. Feyereisen and J. Lannoy, *Gestures and speech: psychological investigations*, ser. Studies in emotion and social interaction. Cambridge University Press, 1991.

[115] J. L. Fleiss, *Statistical methods for rates and proportions*, ser. Wiley series in probability and mathematical statistics. New York, Chichester, Toronto: Wiley, cop1981, 1981.

[116] N. F. Fragopanagos and J. G. Taylor, "Emotion recognition in human-computer interaction," *Neural Networks*, vol. 18, no. 4, pp. 389–405, 2005.

[117] D. J. France, R. G. Shiavi, S. E. Silverman, M. K. Silverman, and D. M. Wilkes, "Acoustical properties of speech as indicators of depression and suicidal risk," *Biomedical Engineering*, vol. 47, no. 7, pp. 829–837, 2000.

[118] A. J. Fridlund, *Human facial expression: an evolutionary view*. Academic Press, 1994.

[119] N. Frijda, *The Emotions*, ser. Studies in Emotion and Social Interaction. Cambridge University Press, 1986.

[120] H. Gardner, *Frames of Mind: The Theory of Multiple Intelligences*. Basic Books, 2011.

[121] K. Geneva, A. Siegen, and T. Wisconsin, *Appraisal Processes in Emotion : Theory, Methods, Research: Theory, Methods, Research*, ser. SAS Series. Oxford University Press, USA, 2001.

[122] D. Gerhard, "Pitch extraction and fundamental frequency: History and current techniques," Department of Computer Science, University of Regina, Regina, Saskatchewan, Canada, Technical Report TR-CS 2003-06, November 2003.

[123] S. W. Gilroy, M. Cavazza, R. Chaignon, S.-M. Mäkelä, M. Niranen, E. André, T. Vogt, J. Urbain, M. Billinghurst, H. Seichter, and M. Benayoun, "E-tree: Emotionally driven augmented reality art," in *International Conference on Multimedia (MM)*, New York, NY, USA, 2008, pp. 945–948.

[124] S. W. Gilroy, M. Cavazza, M. Niranen, E. Andre, T. Vogt, J. Urbain, M. Benayoun, H. Seichter, and M. Billinghurst, "PAD-based multimodal affective fusion," in *International Conference on Affective Computing and Intelligent Interaction (ACII)*, 2009.

[125] M. Glodek, M. Schels, G. Palm, and F. Schwenker, "Multi-modal fusion based on classifiers using reject options and markov fusion networks," in *International Conference on Pattern Recognition (ICPR)*, Nov 2012, pp. 1084–1087.

[126] M. Glodek, F. Honold, T. Geier, G. Krell, F. Nothdurft, S. Reuter, F. Schüssel, T. Hörnle, K. Dietmayer, W. Minker, S. Biundo, M. Weber, G. Palm, and F. Schwenker, "Fusion

paradigms in cognitive technical systems for human-computer interaction," *Neurocomputing*, vol. 161, pp. 17 – 37, 2015.

[127] D. Goleman, *Emotional Intelligence: Why It Can Matter More Than IQ*. Bantam, June 1997.

[128] K. Green, "Studies of the McGurk effect: implications for theories of speech perception," in *International Conference on Spoken Language Processing (ICSLP)*, October 1996, pp. 1652–1655vol.3.

[129] M. Grewal and A. Andrews, "Applications of Kalman filtering in aerospace 1960 to the present [historical perspectives]," *Control Systems*, vol. 30, no. 3, pp. 69–78, June 2010.

[130] M. Grimm and K. Kroschel, *Emotion Estimation in Speech Using a 3D Emotion Space Concept*. INTECH Open Access Publisher, 2007.

[131] M. Grimm, K. Kroschel, and S. Narayanan, "The vera am mittag german audio-visual emotional speech database," in *International Conference on Multimedia and Expo (ICME)*, April 2008, pp. 865–868.

[132] M. Grimm, K. Kroschel, E. Mower, and S. Narayanan, "Primitives-based evaluation and estimation of emotions in speech," *Speech Communication*, vol. 49, no. 10-11, pp. 787–800, October 2007.

[133] H. Gunes and M. Pantic, "Automatic, dimensional and continuous emotion recognition," *Synthetic Emotion*, vol. 1, no. 1, pp. 68–99, 2010.

[134] H. Gunes and M. Piccardi, "Affect recognition from face and body: early fusion vs. late fusion," in *International Conference on Systems, Man and Cybernetics*, 2005, pp. 3437–3443Vol. 4.

[135] H. Gunes, M. Piccardi, and T. Jan, "Face and body gesture recognition for a vision-based multimodal analyzer," in *European Workshop on Visual Information Processing (EUVIP)*, Darlinghurst, Australia, Australia, 2004, pp. 19–28.

[136] E. A. Haggard and K. S. Isaacs, *Micromomentary facial expressions as indicators of ego mechanisms in psychotherapy*. Appleton-Century-Crofts, 1966, pp. 154–165.

[137] D. Hall and J. Llinas, "An introduction to multisensor data fusion," *Proceedings of the IEEE*, vol. 85, no. 1, pp. 6–23, 1997.

[138] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA data mining software: an update," *SIGKDD Explorations Newsletter*, vol. 11, no. 1, pp. 10–18, November 2009.

[139] K. Han, D. Yu, and I. Tashev, "Speech emotion recognition using deep neural network and extreme learning machine," in *Conference of the International Speech Communication Association (INTERSPEECH)*, September 2014.

[140] A. Hanjalic, "Extracting moods from pictures and sounds: towards truly personalized TV," *IEEE Signal Processing Magazine*, vol. 23, no. 2, pp. 90–100, March 2006.

[141] S. Hantke, T. Appel, F. Eyben, and B. Schuller, "iHEARu-PLAY: Introducing a game for crowdsourced data collection for affective computing," in *International Conference on Affective Computing and Intelligent Interaction (ACII)*, 2015.

[142] H. Hariharan, A. Koschan, B. R. Abidi, A. Gribok, and M. A. Abidi, "Fusion of visible and infrared images using empirical mode decomposition to improve face recognition." in *International Conference on Image Processing (ICIP)*, 2006, pp. 2049–2052.

[143] B. M. Hart, T. G. J. Abresch, and W. Einhäuser, "Faces in places: Humans and machines make similar face detection errors," *PLoS One*, vol. 6(10):e25373, 2011.

[144] J. Henrich, S. J. Heine, and A. Norenzayan, "The weirdest people in the world?" *Behavioral and Brain Sciences*, vol. 33, no. 2-3, pp. 61–83, June 2010.

[145] J. Hofmann, F. Stoffel, A. Weber, and T. Platt, "The 16 enjoyable emotions induction task (16-EEIT)," 2012, unpublished.

[146] L. Hoste, B. Dumas, and B. Signer, "Mudra: A unified multimodal interaction framework," in *International Conference on Multimodal Interfaces (ICMI)*, New York, NY, USA, 2011, pp. 97–104.

[147] H. Hung and G. Chittaranjan, "The idiap wolf corpus: exploring group behaviour in a competitive role-playing game." in *International Conference on Multimedia (MM)*, 2010, pp. 879–882.

[148] H. Hung and D. Gatica-Perez, "Estimating cohesion in small groups using audio-visual nonverbal behavior." *Multimedia*, vol. 12, no. 6, pp. 563–575, 2010.

[149] H. Hung, D. B. Jayagopi, C. Yeo, G. Friedland, S. O. Ba, J. Odobez, K. Ramchandran, N. Mirghafori, and D. Gatica-Perez, "Using audio and video features to classify the most dominant person in a group meeting," in *International Conference on Multimedia (MM)*, 2007, pp. 835–838.

[150] A. M. Isen, K. A. Daubman, and G. P. Nowicki, "Positive affect facilitates creative problem solving," *Journal of Personality and Social Psychology*, vol. 52, no. 6, pp. 1122–1131, 1987.

[151] A. Isen, B. Means, R. Patrick, and G. Nowicki, "Some factors influencing decision making strategy and risk taking," *Affect and Cognition*, pp. 243–261, 1982.

[152] A. Jameel, A. Ghafoor, and M. M. Riaz, "Improved guided image fusion for magnetic resonance and computed tomography imaging," *The Scientific World Journal*, vol. 2014, no. 7, pp. 1–7, 2014.

[153] A. P. James and B. V. Dasarathy, "Medical image fusion: A survey of the state of the art," *Information Fusion*, vol. 19, no. 0, pp. 4–19, 2014, special Issue on Information Fusion in Medical Image Computing and Systems.

[154] W. James, "What is an emotion?" *Mind*, vol. 9, no. 34, pp. 188–205, 1884.

[155] D. B. Jayagopi, H. Hung, C. Yeo, and D. Gatica-Perez, "Modeling dominance in group conversations using nonverbal activity cues," *Audio, Speech and Language Processing*, vol. 17, no. 3, pp. 501–513, March 2009.

[156] M. Johnston, "Multimodal language processing." in *International Conference on Spoken Language Processing (ICSLP)*, 1998.

[157] M. Johnston and S. Bangalore, "Finite-state multimodal parsing and understanding." in *International Conference on Computational Linguistics (COLING)*, 2000, pp. 369–375.

[158] M. Johnston, P. R. Cohen, D. McGee, S. L. Oviatt, J. A. Pittman, and I. Smith, "Unification-based multimodal integration," in *Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics*, Stroudsburg, PA, USA, 1997, pp. 281–288.

[159] T. Johnstone, "Emotional speech elicited using computer games," in *International Conference on Spoken Language Processing (ICSLP)*, 1996.

[160] T. Johnstone and K. R. Scherer, "The effects of emotions on voice quality," in *International Congress of Phonetic Sciences*, San Francisco, August 1999, pp. 2029–2032.

[161] N. Jones, "Computer science: The learning machines," *Nature*, vol. 505, no. 7482, pp. 146–148, January 2014.

[162] S. E. Kahou, C. Pal, X. Bouthillier, P. Froumenty, c. Gülçehre, R. Memisevic, P. Vincent, A. Courville, Y. Bengio, R. C. Ferrari, M. Mirza, S. Jean, P.-L. Carrier, Y. Dauphin, N. Boulanger-Lewandowski, A. Aggarwal, J. Zumer, P. Lamblin, J.-P. Raymond, G. Desjardins, R. Pascanu, D. Warde-Farley, A. Torabi, A. Sharma, E. Bengio, M. Côté, K. R. Konda, and Z. Wu, "Combining modality specific deep neural networks for emotion recognition in video," in *International Conference on Multimodal Interaction (ICMI)*, New York, NY, USA, 2013, pp. 543–550.

[163] S. E. Kahou, X. Bouthillier, P. Lamblin, Ç. Gülçehre, V. Michalski, K. R. Konda, S. Jean, P. Froumenty, Y. Dauphin, N. Boulanger-Lewandowski, R. C. Ferrari, M. Mirza, D. Warde-Farley, A. C. Courville, P. Vincent, R. Memisevic, C. J. Pal, and Y. Bengio, "EmoNets: Multimodal deep learning approaches for emotion recognition in video," *CoRR*, vol. abs/1503.01800, 2015.

[164] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960.

[165] I. Kanluan, M. Grimm, and K. Kroschel, "Audio-visual emotion recognition using an emotion space concept," in *European Signal Processing Conference (EUSIPCO)*, August 2008.

[166] K. Karpouzis, G. Caridakis, L. Kessous, N. Amir, A. Raouzaiou, L. Malatesta, and S. Kollias, "Modeling naturalistic affective states via facial, vocal, and bodily expressions recognition," in *International Conference on Artifical Intelligence for Human Computing*, Berlin, Heidelberg, 2007, pp. 91–112.

[167] J. F. Kelley, "An iterative design methodology for user-friendly natural language office information applications," *Information Systems*, vol. 2, no. 1, pp. 26–41, January 1984.

[168] A. Kendon, "Some functions of gaze-direction in social interaction," *Acta Psychol (Amst)*, vol. 26, no. 1, pp. 22–63, 1967.

[169] P. M. Kenealy, "The velten mood induction procedure: A methodological review," *Motivation and Emotion*, vol. 10, no. 4, pp. 315–335, December 1986.

[170] J. Kim, "Bimodal emotion recognition using speech and physiological changes," in *Robust Speech Recognition and Understanding*, M. Grimm and K. Kroschel, Eds. I-Tech Education and Publishing, Vienna, Austria, 2007, pp. 265–280.

[171] J. Kim and F. Lingenfelser, "Ensemble approaches to parametric decision fusion for bimodal emotion recognition," in *International Conference on Bio-inspired Systems and Signal Processing (BIOSIGNALS)*, 2010, pp. 460–463.

[172] M. Kipp, "ANVIL - a generic annotation tool for multimodal dialogue." in *Conference of the International Speech Communication Association (INTERSPEECH)*, 2001, pp. 1367–1370.

[173] M. Kipp, "ANVIL: The video annotation research tool," in *Handbook of Corpus Phonology*. Oxford, UK: Oxford University Press, 2013.

[174] G. Klasmeyer and W. Sendlmeier, "The classification of different phonation types in emotional and neutral speech," *Speech Language and the Law*, vol. 4, no. 1, 1997.

[175] A. Kleinsmith and N. Bianchi-Berthouze, "Recognizing affective dimensions from body posture," in *Affective Computing and Intelligent Interaction*, ser. Lecture Notes in Computer Science, A. Paiva, R. Prada, and R. Picard, Eds. Springer Berlin Heidelberg, 2007, vol. 4738, pp. 48–58.

[176] M. L. Knapp and J. A. Hall, *Nonverbal Communication in Human Interaction*, R. A. Hinde, Ed. Wadsworth: Thomas Learning, 2007.

[177] H. Kobayashi and S. Kohshima, "Evolution of the human eye as a device for communication," in *Primate Origins of Human Cognition and Behavior*, T. Matsuzawa, Ed. Springer Japan, 2001, pp. 383–401.

[178] S. G. Kong, J. Heo, F. Boughorbel, Y. Zheng, B. R. Abidi, A. Koschan, M. Yi, and M. A. Abidi, "Multiscale fusion of visible and thermal IR images for illumination-invariant face recognition," *Int. J. Comput. Vision*, vol. 71, no. 2, pp. 215–233, February 2007.

[179] R. M. Krauss, Y. Chen, and P. Chawla, "Nonverbal behavior and nonverbal communication: What do conversational hand gestures tell us?" *Advances in Experimental Social Psychology*, vol. 28, pp. 389–450, 1996.

[180] R. E. Kraut and R. E. Johnston, "Social and emotional messages of smiling: An ethological approach." *Personality and Social Psychology*, vol. 37, no. 9, pp. 1539–1553, 1979.

[181] K. Krishna Kishore and P. Krishna Satish, "Emotion recognition in speech using MFCC and wavelet features," in *International Conference on Advance Computing Conference (IACC)*, February 2013, pp. 842–847.

[182] P. O. Kristensson and L. C. Denby, "Continuous recognition and visualization of pen strokes and touch-screen gestures," in *Eurographics Symposium on Sketch-Based Interfaces and Modeling*, New York, NY, USA, 2011, pp. 95–102.

[183] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105.

[184] J. L. Lakin, V. E. Jefferis, C. M. Cheng, and T. L. Chartrand, "The chameleon effect as social glue: Evidence for the evolutionary significance of nonconscious mimicry," *Nonverbal Behavior*, vol. 27, no. 3, pp. 145+, 2003.

[185] D. Lalanne, L. Nigay, p. Palanque, P. Robinson, J. Vanderdonckt, and J.-F. Ladry, "Fusion engines for multimodal input: A survey," in *International Conference on Multimodal Interfaces (ICMI)*, New York, NY, USA, 2009, pp. 153–160.

[186] P. J. Lang, M. M. Bradley, and B. N. Cuthbert, "International affective picture system (iaps): Affective ratings of pictures and instruction manual," The Center for Research in Psychophysiology, University of Florida, Gainesville, FL, Technical Report A-8, 2008.

[187] M. E. Latoschik, "Designing transition networks for multimodal vr-interactions using a markup language," in *International Conference on Multimodal Interfaces (ICMI)*, 2002, pp. 411–416.

[188] Q. V. Le, "Building high-level features using large scale unsupervised learning," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2013, pp. 8595–8598.

[189] C. M. Lee, S. Yildirim, M. Bulut, A. Kazemzadeh, C. Busso, Z. Deng, S. Lee, and S. Narayanan, "Emotion recognition based on phoneme classes," in *International Conference on Spoken Language Processing (ICSLP)*, 2004, pp. 889–892.

[190] S. Lee, S. Yildirim, A. Kazemzadeh, and S. Narayanan, "An articulatory study of emotional speech production." in *Conference of the International Speech Communication Association (INTERSPEECH)*, 2005, pp. 497–500.

[191] S. Lee, E. Bresch, J. Adams, A. Kazemzadeh, and S. Narayanan, "A study of emotional speech articulation using a fast magnetic resonance imaging technique." in *Conference of the International Speech Communication Association (INTERSPEECH)*, 2006.

[192] Y. Leng, X. Xu, and G. Qi, "Combining active learning and semi-supervised learning to construct svm classifier," *Knowledge-Based Systems*, vol. 44, pp. 121–131, May 2013.

[193] L. Li, Y. Zhao, D. Jiang, Y. Zhang, F. Wang, I. Gonzalez, E. Valentin, and H. Sahli, "Hybrid deep neural network–hidden Markov model (dnn-HMM) based speech emotion recognition," in *International Conference on Affective Computing and Intelligent Interaction (ACII)*, Washington, DC, USA, 2013, pp. 312–317.

[194] X. Li, J. Tao, M. Johnson, J. Soltis, A. Savage, K. Leong, and J. Newman, "Stress and emotion classification using jitter and shimmer features," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2007, pp. IV–1081–IV–1084.

[195] J. Lichtenauer, J. Shen, M. F. Valstar, and M. Pantic, "Cost-effective solution to synchronised audio-visual data capture using multiple sensors," *Image and Vision Computing*, vol. 29, pp. 666–680, September 2011.

[196] R. Lienhart and J. Maydt, "An extended set of Haar-like features for rapid object detection," in *International Conference on Image and Signal Processing (ICISP)*, 2002, pp. I–900–I–903 vol.1.

[197] F. Lingenfelser, J. Wagner, and E. André, "A systematic discussion of fusion techniques for multi-modal affect recognition tasks," in *International Conference on Multimodal Interfaces (ICMI)*, New York, NY, USA, 2011, pp. 19–26.

[198] F. Lingenfelser, J. Wagner, E. André, G. McKeown, and W. Curran, "An event driven fusion approach for enjoyment recognition in real-time," in *International Conference on Multimedia (MM)*, New York, NY, USA, 2014, pp. 377–386.

[199] M. Mancini, L. Ach, E. Bantegnie, T. Baur, N. Berthouze, D. Datta, Y. Ding, S. Dupont, H. Griffin, F. Lingenfelser, R. Niewiadomski, C. Pelachaud, O. Pietquin, B. Piot, J. Urbain, G. Volpe, and J. Wagner, "Laugh when you're winning," in *Innovative and Creative Developments in Multimodal Interaction Systems*, ser. IFIP Advances in Information and Communication Technology, Y. Rybarczyk, T. Cardoso, J. Rosas, and L. Camarinha-Matos, Eds. Springer Berlin Heidelberg, 2014, vol. 425, pp. 50–79.

[200] I. Marsic, A. Medl, and J. Flanagan, "Natural communication with information systems," *Proceedings of the IEEE*, vol. 88, no. 8, pp. 1354–1366, 2000.

[201] H. P. Martinez, Y. Bengio, and G. N. Yannakakis, "Learning deep physiological models of affect," *Computational Intelligence Magazine, IEEE*, vol. 8, no. 2, pp. 20–33, May 2013.

[202] G. McKeown, M. Valstar, R. Cowie, and M. Pantic, "The SEMAINE corpus of emotionally coloured character interactions," in *International Conference on Multimedia and Expo (ICME)*, July 2010, pp. 1079–1084.

[203] G. McKeown, W. Curran, J. Wagner, F. Lingenfelser, and E. André, "The belfast storytelling database – a spontaneous social interaction database with laughter focused annotation," in *International Conference on Affective Computing and Intelligent Interaction and Workshops (ACII)*, Xi'an, China, September 2015.

[204] S. Meduri and R. Ananth, *A Survey and Evaluation of Voice Activity Detection Algorithms: Speech Processing Module*. Germany: LAP Lambert Academic Publishing, 2012.

[205] G. Mehlmann, B. Endraß, and E. André, "Modeling parallel state charts for multithreaded multimodal dialogues," in *International Conference on Multimodal Interfaces (ICMI)*, New York, NY, USA, 2011, pp. 385–392.

[206] G. U. Mehlmann and E. André, "Modeling multimodal integration with event logic charts," in *International Conference on Multimodal Interaction (ICMI)*, New York, NY, USA, 2012, pp. 125–132.

[207] A. Mehrabian, *Nonverbal communication*. Aldine Transaction, 1972.

[208] A. Mehrabian and S. R. Ferris, "Inference of attitudes from nonverbal communication in two channels." *Consulting Psychology*, vol. 31, no. 3, pp. 248–252, June 1967.

[209] A. Mehrabian, "Some referents and measures of nonverbal behavior," *Behavior Research Methods*, vol. 1, pp. 203–207, 1969.

[210] R. Meir and G. Rätsch, *Advanced Lectures on Machine Learning*, S. Mendelson and A. J. Smola, Eds. New York, NY, USA: Springer-Verlag New York, Inc., 2003.

[211] D. S. Messinger, M. H. Mahoor, S.-M. Chow, and J. F. Cohn, "Automated measurement of facial expression in infant-mother interaction: A pilot study," *Infancy*, vol. 14, no. 3, pp. 285–305, 2009.

[212] D. Messinger, T. Cassel, S. Acosta, Z. Ambadar, and J. Cohn, "Infant smiling dynamics and perceived positive emotion," *Nonverbal Behavior*, vol. 32, no. 3, pp. 133–155, 2008.

[213] A. Metallinou and S. Narayanan, "Annotation and processing of continuous emotional attributes: Challenges and opportunities," in *International Conference on Automatic Face and Gesture Recognition (FGR)*, April 2013, pp. 1–8.

[214] A. Metallinou, A. Katsamanis, and S. S. Narayanan, "Tracking continuous emotional trends of participants during affective dyadic interactions using body language and speech information," *Image and Vision Computing*, vol. 31, no. 2, pp. 137–152, February 2013.

[215] H. B. Mitchell, *Multi-Sensor Data Fusion: An Introduction*, 1st ed. Springer Publishing Company, Incorporated, 2007.

[216] T. M. Mitchell, *Machine Learning*, 1st ed. New York, NY, USA: McGraw-Hill, Inc., 1997.

[217] A. Mohamed, T. Sainath, G. Dahl, B. Ramabhadran, G. Hinton, and M. Picheny, "Deep belief networks using discriminative features for phone recognition," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2011, pp. 5060–5063.

[218] A. Mohamed, G. E. Dahl, and G. E. Hinton, "Deep belief networks for phone recognition," in *NIPS Workshop on Deep Learning for Speech Recognition and Related Applications*, 2009.

[219] E. Mower, A. Metallinou, C.-C. Lee, A. Kazemzadeh, C. Busso, S. Lee, and S. Narayanan, "Interpreting ambiguous emotional expressions," in *International Conference on Affective Computing and Intelligent Interaction (ACII)*, September 2009, pp. 1–8.

[220] I. R. Murray and J. L. Arnott, "Toward the simulation of emotion in synthetic speech: A review of the literature on human vocal emotion," *Acoustical Society of America*, vol. 93, no. 2, pp. 1097–1108, 1993.

[221] C. Nass and S. Brave, *Wired for Speech: How Voice Activates and Advances the Human-Computer Relationship*. Cambridge, MA: MIT Press, 2005.

[222] D. Navarre, P. A. Palanque, R. Bastide, A. Schyn, M. Winckler, L. P. Nedel, and C. M. D. S. Freitas, "A formal description of multimodal interaction techniques for immersive virtual reality applications." in *International Conference on Human-Computer Interaction (INTERACT)*, 2005, pp. 170–183.

[223] D. Navarre, "Posture sharing in dyadic interaction," *American Journal of Dance Therapy*, vol. 5, pp. 28–42, 1982.

[224] D. Neiberg, K. Elenius, and K. Laskowski, "Emotion recognition in spontaneous speech using GMMs," in *Conference of the International Speech Communication Association (INTERSPEECH)*, 2006.

[225] M. A. Nicolaou, H. Gunes, and M. Pantic, "Audio-visual classification and fusion of spontaneous affective data in likelihood space." in *International Conference on Pattern Recognition (ICPR)*, 2010, pp. 3695–3699.

[226] M. A. Nicolaou, H. Gunes, and M. Pantic, "Continuous prediction of spontaneous affect from multiple cues and modalities in valence-arousal space." *Affective Computing*, vol. 2, no. 2, pp. 92–105, 2011.

[227] R. Niewiadomski and C. Pelachaud, "Towards multimodal expression of laughter." in *International Conference on Intelligent Virtual Agents (IVA)*, 2012, pp. 231–244.

[228] R. Niewiadomski, M. Mancini, T. Baur, G. Varni, H. Griffin, and M. Aung, "Mmli: Multimodal multiperson corpus of laughter in interaction," in *Human Behavior Understanding*, ser. Lecture Notes in Computer Science, A. Salah, H. Hung, O. Aran, and H. Gunes, Eds. Springer International Publishing, 2013, vol. 8212, pp. 184–195.

[229] L. Nigay and J. Coutaz, "A design space for multimodal systems: Concurrent processing and data fusion," in *International Conference on Human-Computer Interaction (INTERACT) and International Conference on Human Factors in Computing Systems (CHI)*, New York, NY, USA, 1993, pp. 172–178.

[230] D. O. Olguín, P. A. Gloor, and A. S. Pentland. (2009) Capturing individual and group behavior with wearable sensors.

[231] C. E. Osgood, W. H. May, and M. S. Miron, *Cross-cultural universals of affective meaning*. Urbana: University of Illinois Press, 1975.

[232] K. Otsuka, Y. Takemae, and J. Yamato, "A probabilistic inference of multiparty-conversation structure based on Markov-switching models of gaze patterns, head

directions, and utterances," in *International Conference on Multimodal Interfaces (ICMI)*, New York, NY, USA, 2005, pp. 191–198.

[233] S. Oviatt, A. DeAngeli, and K. Kuhn, "Integration and synchronization of input modes during multimodal human-computer interaction," in *Referring Phenomena in a Multimedia Context and Their Computational Treatment*, Stroudsburg, PA, USA, 1997, pp. 1–13.

[234] S. Oviatt, P. Cohen, L. Wu, J. Vergo, L. Duncan, B. Suhm, J. Bers, T. Holzman, T. Winograd, J. Landay, J. Larson, and D. Ferro, "Designing the user interface for multimodal speech and pen-based gesture applications: State-of-the-art systems and future research directions," *Human-Computer Interaction*, vol. 15, no. 4, pp. 263–322, December 2000.

[235] P. Pal, A. Iyer, and R. Yantorno, "Emotion detection from infant facial expressions and cries," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2006, pp. II–II.

[236] S. J. Pan and Q. Yang, "A survey on transfer learning," *Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, October 2010.

[237] M. Pantic and L. J. M. Rothkrantz, "Toward an affect-sensitive multimodal human-computer interaction," *Proceedings of the IEEE*, vol. 91, no. 9, pp. 1370–1390, September 2003.

[238] M. Pantic, A. Pentland, A. Nijholt, and T. S. Huang, "Human computing and machine understanding of human behavior: A survey," in *Artifical Intelligence for Human Computing, ICMI 2006 and IJCAI 2007 International Workshops, Banff, Canada, November 3, 2006, Hyderabad, India, January 6, 2007, Revised Seleced and Invited Papers*, 2007, pp. 47–71.

[239] M. Pantic, A. Nijholt, A. Pentland, and T. S. Huang, "Human-centred intelligent human-computer interaction (HCI2): how far are we from attaining it?" *International Journal of Autonomous and Adaptive Communications Systems (IJAACS)*, vol. 1, no. 2, pp. 168–187, 2008.

[240] M. Pantic, R. Cowie, F. D'Errico, D. Heylen, M. Mehu, C. Pelachaud, I. Poggi, M. Schröder, and A. Vinciarelli, "Social signal processing: The research agenda," in *Visual Analysis of Humans - Looking at People.* Springer, 2011, pp. 511–538.

[241] A. Pentland, "Social dynamics: Signals and behavior," in *International Conference on Development and Learning (ICDL)*, La Jolla, CA, October 2004.

[242] A. Pentland, "Social signal processing," *IEEE Signal Processing Magazine*, vol. 24, no. 4, pp. 108–111, 2007.

[243] A. Pentland, "Socially aware computation and communication," *Computer*, vol. 38, no. 3, pp. 33–40, 2005.

[244] A. Pentland, *Honest Signals: How They Shape Our World*. The MIT Press, 2008.

[245] S. Petridis and M. Pantic, "Audiovisual discrimination between laughter and speech," in *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on*, 2008, pp. 5117–5120.

[246] T. Pfister and P. Robinson, "Real-time recognition of affective states from nonverbal features of speech and its application for public speaking skill analysis," *Affective Computing*, vol. 2, no. 2, pp. 66–78, 2011.

[247] F. Pianesi, M. Zancanaro, E. Not, C. Leonardi, V. Falcon, and B. Lepri, "Multimodal support to group dynamics," *Personal Ubiquitous Computing*, vol. 12, no. 3, pp. 181–195, January 2008.

[248] R. W. Picard, *Affective computing*. Cambridge, MA, USA: MIT Press, 1997.

[249] J. Platt, *Advances in Kernel Methods: Support Vector Learning*. Cambridge, MA: MIT press, 1999, ch. Fast training of SVMs using sequential minimal optimization, pp. 185–208.

[250] R. Plutchik, *A general psychoevolutionary theory of emotion*. New York: Academic press, 1980, pp. 3–33.

[251] I. Poggi and F. D'Errico, "Dominance signals in debates," in *Human Behavior Understanding*, ser. Lecture Notes in Computer Science, A. Salah, T. Gevers, N. Sebe, and A. Vinciarelli, Eds. Springer Berlin Heidelberg, 2010, vol. 6219, pp. 163–174.

[252] I. Poggi and D. Francesca, "Cognitive modelling of human social signals," in *International Workshop on Social Signal Processing*, New York, NY, USA, 2010, pp. 21–26.

[253] R. Polikar, "Ensemble based systems in decision making," *Circuits and Systems Magazine, IEEE*, vol. 6, no. 3, pp. 21–45, Third 2006.

[254] F. E. Pollick, H. M. Paterson, A. Bruderlin, and A. J. Sanford, "Perceiving affect from arm movement," *Cognition*, vol. 82, no. 2, pp. B51 – B61, 2001.

[255] D. M. W. Powers, "Evaluation: From precision, recall and F-factor to ROC, informedness, markedness & correlation," School of Informatics and Engineering, Flinders University, Adelaide, Australia, Technical Report SIE-07-001, 2007.

[256] J. Proakis and D. Manolakis, *Digital signal processing*. Pearson Prentice Hall, 2007.

[257] L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, February 1989.

[258] F. H. Rauscher, R. M. Krauss, and Y. Chen, "Gesture, speech, and lexical access: The role of lexical movements in speech prolexical," *Psychological Science*, vol. 7, no. 4, pp. 226–231, 1996.

[259] P. Read, M. Meyer, and G. Group, *Restoration of Motion Picture Film*, ser. Butterworth-Heinemann series in conservation and museology. Butterworth-Heinemann, 2000.

[260] R. A. Redner and H. F. Walker, "Mixture densities, maximum likelihood and the em algorithm," *SIAM Review*, vol. 26, no. 2, pp. 195–239, 1984.

[261] V. Richmond and J. McCroskey, *Nonverbal behavior in interpersonal relations*. Allyn and Bacon, 1995.

[262] R. Rienks, D. Zhang, D. Gatica-Perez, and W. Post, "Detection and application of influence rankings in small group meetings," in *International Conference on Multimodal Interfaces (ICMI)*, New York, November 2006, pp. 257–264.

[263] S. Rifai, Y. Bengio, A. C. Courville, P. Vincent, and M. Mirza, "Disentangling factors of variation for facial expression recognition," in *European Conference on Computer Vision (ECCV)*, 2012, pp. 808–822.

[264] B. Rimé, "The elimination of visible behaviour from social interactions: Effects on verbal, nonverbal and interpersonal variables," *European Journal of Social Psychology*, vol. 12, pp. 113–129, 1982.

[265] B. Rimé, L. Schiaratura, M. Hupet, and A. Ghysselinckx, "Effects of relative immobilization on the speaker's nonverbal behavior and on the dialogue imagery level," *Motivation and Emotion*, vol. 8, pp. 311–325, 1984.

[266] W. Ruch and P. Ekman, "The expressive pattern of laughter," *Emotion qualia, and consciousness*, pp. 426–443, 2001.

[267] A. A. Salah, M. Pantic, and A. Vinciarelli, "Recent developments in social signal processing." in *International Conference on Systems, Man and Cybernetics (SMC)*, 2011, pp. 380–385.

[268] P. Salovey and D. Grewal, "The science of emotional intelligence," *Current Directions in Psychological Science*, vol. 14, no. 6, pp. 281–285, December 2005.

[269] P. Salovey and J. D. Mayer, "Emotional intelligence," *Imagination, Cognition and Personality*, vol. 9, no. 3, pp. 185–211, January 1989.

[270] J. Sasiadek and P. Hartana, "Sensor data fusion using Kalman filter," in *International Conference on Information Fusion (FUSION)*, July 2000, pp. WED5/19–WED5/25 vol.2.

[271] A. E. Scheflen, "The significance of posture in communication systems," *Psychiatry*, vol. 27, pp. 316–331, November 1964.

[272] K. R. Scherer, "Speech and emotional states," *Speech Evaluation in Psychiatry*, pp. 189–220, 1981.

[273] K. R. Scherer, *On the nature and function of emotion: A component process approach.* Hillsdale, NJ: Lawrence Erlbaum, 1984, ch. 14, pp. 293–317.

[274] K. R. Scherer, "Expression of emotion in voice and music," *Voice*, vol. 9, pp. 235–248, 1995.

[275] K. R. Scherer, *Appraisal theory*. Chichester, U.K.: Wiley, 1999, pp. 637–663.

[276] K. R. Scherer, "Vocal communication of emotion: a review of research paradigms," *Speech Communication*, vol. 40, pp. 227–256, April 2003.

[277] K. R. Scherer and M. R. Zentner, "Emotional effects of music: Production rules," in *Music and emotion: Theory and research*. Oxford University Press, 2001, pp. 361–392.

[278] S. Scherer, G. Stratou, M. Mahmoud, J. Boberg, J. Gratch, A. Rizzo, and L.-P. Morency, "Automatic behavior descriptors for psychological disorder analysis," in *International Conference on Automatic Face and Gesture Recognition (FGR)*, 2013.

[279] S. Scherer, M. Glodek, F. Schwenker, N. Campbell, and G. Palm, "Spotting laughter in natural multiparty conversations: A comparison of automatic online and offline approaches using audiovisual data," *Interaction Intelligent System*, vol. 2, no. 1, pp. 4:1–4:31, March 2012.

[280] F. Schiel, S. Steininger, and U. Türk, "The smartkom multimodal corpus at BAS," in *International Conference on Language Resources and Evaluation (LREC)*, 2002.

[281] A. Schlögl, "Overview of data formats for biomedical signals." in *Image Processing, Biosignal Processing, Modelling and Simulation, Biomechanics*, 2009, pp. 1557–1560.

[282] H. Schlosberg, "Three dimensions of emotion," *Psychology Review*, vol. 61, pp. 81–88, 1954.

[283] T. Schmidt, "Transcribing and annotating spoken language with EXMARaLDA," in *International Conference on Language Resources and Evaluation (LREC) Workshop on XML based richly annotated corpora*, Paris, 2004, pp. 879–896, eN.

[284] M. Schröder, E. Bevacqua, R. Cowie, F. Eyben, H. Gunes, D. Heylen, M. ter Maat, G. McKeown, S. Pammi, M. Pantic, C. Pelachaud, B. Schuller, E. de Sevin, M. Valstar, and M. Wollmer, "Building autonomous sensitive artificial listeners," *Affective Computing*, vol. 3, no. 2, pp. 165–183, 2012.

[285] M. Schröder, R. Cowie, E. Douglas-Cowie, S. Savvidou, E. McMahon, and M. Sawey, "FEELTRACE: An instrument for recording perceived emotion in real time," in *ISCA Workshop on Speech and Emotion: A Conceptual Framework for Research*, Belfast, 2000, pp. 19–24.

[286] M. Schröder, P. Baggia, F. Burkhardt, C. Pelachaud, C. Peter, and E. Zovato, "EmotionML – an upcoming standard for representing emotions and related states," in *International Conference on Affective Computing and Intelligent Interaction (ACII)*, 2011, pp. 316–325.

[287] B. Schuller, D. Arsic, G. Rigoll, M. Wimmer, and B. Radig, "Audiovisual behavior modeling by combined feature spaces," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, April 2007, pp. II–733–II–736.

[288] B. Schuller, "Voice and speech analysis in search of states and traits." in *Computer Analysis of Human Behavior*, A. A. Salah and T. Gevers, Eds. Springer, 2011, pp. 227–253.

[289] B. Schuller, R. Müller, F. Eyben, J. Gast, B. Hörnler, M. Wöllmer, G. Rigoll, A. Höthker, and H. Konosu, "Being bored? Recognising natural interest by extensive audiovisual integration for real-life application." *Image Vision Computing*, vol. 27, no. 12, pp. 1760–1774, 2009.

[290] B. Schuller, S. Steidl, and A. Batliner, "The INTERSPEECH 2009 emotion challenge," in *Conference of the International Speech Communication Association (INTERSPEECH)*, 2009, pp. 312–315.

[291] B. Schuller, M. Wöllmer, T. Moosmayr, and G. Rigoll, "Recognition of noisy speech: A comparative survey of robust model architecture and feature enhancement," *Audio Speech Music Processing*, vol. 2009, pp. 5:1–5:17, January 2009.

[292] B. Schuller, S. Steidl, A. Batliner, F. Burkhardt, L. Devillers, C. A. Müller, and S. S. Narayanan, "The INTERSPEECH 2010 paralinguistic challenge," in *Conference of the International Speech Communication Association (INTERSPEECH)*, 2010, pp. 2794–2797.

[293] B. Schuller, A. Batliner, S. Steidl, and D. Seppi, "Recognising realistic emotions and affect in speech: State of the art and lessons learnt from the first challenge," *Speech Communication*, vol. 53, no. 9-10, pp. 1062–1087, 2011.

[294] B. Schuller, S. Steidl, A. Batliner, F. Schiel, and J. Krajewski, "The INTERSPEECH 2011 speaker state challenge," in *Conference of the International Speech Communication Association (INTERSPEECH)*, 2011, pp. 3201–3204.

[295] B. Schuller, S. Steidl, A. Batliner, E. Nöth, A. Vinciarelli, F. Burkhardt, R. Van Son, f. Weninger, F. Eyben, T. Bocklet, G. Mohammadi, and B. Weiss, "The INTERSPEECH 2012 speaker trait challenge," in *Conference of the International Speech Communication Association (INTERSPEECH)*, 2012.

[296] B. Schuller, S. Steidl, A. Batliner, A. Vinciarelli, K. Scherer, F. Ringeval, M. Chetouani, F. Weninger, F. Eyben, E. Marchi, M. Mortillaro, H. Salamin, A. Polychroniou, F. Valente, and S. Kim, "The INTERSPEECH 2013 computational paralinguistics challenge: Social signals, conflict, emotion, autism," in *Conference of the International Speech Communication Association (INTERSPEECH)*, September 2013.

[297] B. Schuller, S. Steidl, A. Batliner, J. Epps, F. Eyben, F. Ringeval, E. Marchi, and Y. Zhang, "The INTERSPEECH 2014 computational paralinguistics challenge: cognitive & physical load," in *Conference of the International Speech Communication Association (INTERSPEECH)*, 2014, pp. 427–431.

[298] N. Sebe, I. Cohen, T. Gevers, and T. S. Huang, "Multimodal approaches for emotion recognition: a survey," in *Internet Imaging VI*, December 2004, pp. 56–67.

[299] D. Seppi, A. Batliner, B. Schuller, S. Steidl, T. Vogt, J. Wagner, L. Devillers, L. Vidrascu, N. Amir, and V. Aharonson, "Patterns, prototypes, performance: classifying emotional user states," in *Conference of the International Speech Communication Association (INTERSPEECH)*, 2008, pp. 601–604.

[300] B. Settles, "Active learning literature survey," University of Wisconsin–Madison, Computer Sciences Technical Report 1648, 2009.

[301] H. A. Simon, "Motivational and emotional controls of cognition." *Psychology Review*, vol. 74, pp. 29–39, 1967.

[302] H. Sloetjes, A. Russel, and A. Klassmann, "ELAN: a free and open-source multimedia annotation tool." in *Conference of the International Speech Communication Association (INTERSPEECH)*, 2007, pp. 4015–4016.

[303] R. Socher, B. Huval, B. Bath, C. D. Manning, and A. Y. Ng, "Convolutional-recursive deep learning for 3D object classification," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 656–664.

[304] M. Song, J. Bu, C. Chen, and N. Li, "Audio-visual based emotion recognition – a new approach," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2004, pp. 1020–1025.

[305] T. Stafylakis, P. Kenny, M. Senoussaoui, and P. Dumouchel, "Preliminary investigation of boltzmann machine classifiers for speaker recognition," in *Odyssey 2012: The Speaker and Language Recognition Workshop, Singapore, June 25-28, 2012*, 2012, pp. 109–116.

[306] S. Steidl, A. Batliner, B. Schuller, and D. Seppi, "The hinterland of emotions: Facing the open-microphone challenge," in *International Conference on Affective Computing and Intelligent Interaction (ACII)*, September 2009, pp. 1–8.

[307] N. L. Stein and K. Oatley, "Basic emotions: Theory and measurement," *Cognition & Emotion*, vol. 6, no. 3–4, pp. 161–168, 1992.

[308] S. S. Stevens, Je, and E. B. Newman, "A scale for the measurement of the psychological magnitude of pitch," *Acoustical Society of America*, vol. 8, pp. 185–190, 1937.

[309] S. Sun, "A survey of multi-view machine learning," *Neural Computing and Applications*, vol. 23, no. 7-8, pp. 2031–2038, 2013.

[310] X. Sun, J. Lichtenauer, M. F. Valstar, A. Nijholt, and M. Pantic, "A multimodal database for mimicry analysis," in *International Conference on Affective Computing and Intelligent Interaction (ACII)*, Memphis, Tennessee, USA, October 2011.

[311] R. Tenney, N. Sandell, M. I. of Technology. Laboratory for Information, and D. Systems, *Detection with Distributed Sensors*, ser. LIDS-P. Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, 1980.

[312] E. L. Thorndike, "A constant error in psychological ratings," *Applied Psychology*, no. 4, pp. 25–29, 1920.

[313] N. Tinbergen, ""Derived" activities; their causation, biological significance, origin, and emancipation during evolution," *Quarterly Review of Biology*, vol. 27, no. 1, pp. 1–32, 1952.

[314] G. Tur, D. Hakkani-Tür, and R. E. Schapire, "Combining active and semi-supervised learning for spoken language understanding," *Speech Communication*, vol. 45, no. 2, pp. 171–186, February 2005.

[315] T. Unterthiner, A. Mayr, G. Klambauer, and S. Hochreiter, "Toxicity prediction using deep learning," *CoRR*, vol. abs/1503.01445, 2015.

[316] J. Urbain and T. Dutoit, "A phonetic analysis of natural laughter, for use in automatic laughter processing systems," in *International Conference on Affective Computing and Intelligent Interaction (ACII)*, 2011, pp. 397–406.

[317] J. Urbain, R. Niewiadomski, E. Bevacqua, T. Dutoit, A. Moinet, C. Pelachaud, B. Picart, J. Tilmanne, and J. Wagner, "AVLaughterCycle: Enabling a virtual agent to join in laughing with a conversational partner using a similarity-driven audiovisual laughter animation," *Multimodal User Interfaces*, vol. 4, no. 1, pp. 47–58, 2010, special Issue: eNTERFACE'09.

[318] J. Urbain, R. Niewiadomski, M. Mancini, H. Griffin, H. Cakmak, L. Ach, and G. Volpe, "Multimodal analysis of laughter for an interactive system," in *Intelligent Technologies for Interactive Entertainment*, ser. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering. Springer International Publishing, 2013, vol. 124, pp. 183–192.

[319] J. Van den Stock, R. Righart, and B. de Gelder, "Body expressions influence recognition of emotions in the face and voice," *Emotion*, vol. 7, no. 3, pp. 487–494, August 2007.

[320] E. Velten, "A laboratory task for induction of mood states," *Behavior Research and Therapy*, vol. 6, pp. 473–482, 1968.

[321] A. Vinciarelli, M. Pantic, H. Bourlard, and A. Pentland, "Social signal processing: State of the art and future perspectives of an emerging domain," in *International Conference on Multimedia (MM)*, Vancouver, Canada, October 2008, pp. 1061–1070.

[322] A. Vinciarelli, A. Dielmann, S. Favre, and H. Salamin, "Canal9: A database of political debates for analysis of social interactions," in *International Conference on Affective Computing and Intelligent Interaction (ACII)*, September 2009, pp. 1–4.

[323] A. Vinciarelli, M. Pantic, D. Heylen, C. Pelachaud, I. Poggi, F. D'ericco, and M. Schroeder, "Bridging the gap between social animal and unsocial machine: A survey of social signal processing," *Affective Computing*, vol. 3, pp. 69–87, April 2012, issue 1.

[324] A. Vinciarelli, M. Pantic, H. Bourlard, and A. Pentland, "Social signals, their function, and automatic analysis: A survey," in *International Conference on Multimodal Interfaces (ICMI)*, New York, NY, USA, 2008, pp. 61–68.

[325] A. Vinciarelli, M. Pantic, and H. Bourlard, "Social signal processing: Survey of an emerging domain," *Image Vision Computing*, vol. 27, no. 12, pp. 1743–1759, November 2009.

[326] P. A. Viola and M. J. Jones, "Rapid object detection using a boosted cascade of simple features," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2001, pp. 511–518.

[327] T. Vogt and E. André, "Comparing feature sets for acted and spontaneous speech in view of automatic emotion recognition," in *International Conference on Multimedia and Expo (ICME)*, July 2005, pp. 474–477.

[328] T. Vogt, E. André, and N. Bee, "EmoVoice – A framework for online recognition of emotions from voice," in *Tutorial and Research Workshop on Perception and Interactive Technologies for Speech-Based Systems: Perception in Multimodal Dialogue Systems*, Berlin, Heidelberg, 2008, pp. 188–199.

[329] T. Waaramaa, P. Alku, and A.-M. Laukkanen, "The role of F3 in the vocal expression of emotions," *Logopedics Phoniatrics Vocology*, vol. 31, no. 4, pp. 153–156, December 2006.

[330] H. L. Wagner and J. Smith, "Facial expression in the presence of friends and strangers," *Nonverbal Behavior*, vol. 15, pp. 201–214, 1991.

[331] J. Wagner, F. Lingenfelser, and E. André, "The social signal interpretation framework (ssi) for real time signal processing and recognition," in *Proc. of Interspeech 2011*, 2011.

[332] J. Wagner, F. Lingenfelser, E. André, J. Kim, and T. Vogt, "Exploring fusion methods for multimodal emotion recognition with missing data," *Affective Computing*, vol. 2, no. 4, pp. 206–218, 2011.

[333] J. Wagner, F. Lingenfelser, N. Bee, and E. André, "Social signal interpretation (ssi) - a framework for real-time sensing of affective and social signals," *KI - KÃ¼nstliche Intelligenz*, vol. 25, pp. 251–256, 2011.

[334] J. Wagner, F. Lingenfelser, and E. André, "A frame pruning approach for paralinguistic recognition tasks," in *Conference of the International Speech Communication Association (INTERSPEECH)*, 2012.

[335] J. Wagner, F. Lingenfelser, and E. André, "Using phonetic patterns for detecting social cues in natural conversations," in *Conference of the International Speech Communication Association (INTERSPEECH)*, 2013, pp. 168–172.

[336] J. Wagner, F. Lingenfelser, T. Baur, I. Damian, F. Kistler, and E. André, "The social signal interpretation (ssi) framework: multimodal signal processing and recognition in real-time," in *International Conference on Multimedia*, New York, NY, USA, 2013, pp. 831–834.

[337] J. Wagner, F. Lingenfelser, and E. André, *Building a Robust System for Multimodal Emotion Recognition*. Wiley, 2015, ch. 15, pp. 379–410.

[338] J. A. Ward, P. Lukowicz, and G. Tröster, "Evaluating performance in continuous context recognition using event-driven error characterisation," in *Location- and Context-Awareness, Second International Workshop (LoCA)*, 2006, pp. 239–255.

[339] J. A. Ward, P. Lukowicz, and H. Gellersen, "Performance metrics for activity recognition," *Intelligent Systems and Technology*, vol. 2, no. 1, p. 6, 2011.

[340] P. Watzlawick and J. Beavin, "Some formal aspects of communication," *American Behavioral Scientist*, vol. 10, pp. 4–8, 1967.

[341] C. E. Williams and K. N. Stevens, "On determining the emotional state of pilots during flight: An exploratory study," *Aerospace Med*, vol. 40, no. 12.1, pp. 1369–1372, December 1969.

[342] E. Williams, "Experimental comparisons of face-to-face and mediated communication: A review," *Psychological Bulletin*, vol. 84, pp. 963–976, September 1977.

[343] T. Wilson, "Annotating subjective content in meetings," in *International Conference on Language Resources and Evaluation (LREC)*, 2008.

[344] J. Wilting, E. Krahmer, and M. Swerts, "Real vs. acted emotional speech." in *INTERSPEECH*, 2006.

[345] P. Wittenburg, H. Brugman, A. Russel, A. Klassmann, and H. Sloetjes, "ELAN: a professional framework for multimodality research," in *International Conference on Language Resources and Evaluation (LREC)*, 2006.

[346] M. Wöllmer, F. Eyben, S. Reiter, B. Schuller, C. Cox, E. Douglas-Cowie, and R. Cowie, "Abandoning emotion classes – towards continuous emotion recognition with modelling of long-range dependencies." in *Conference of the International Speech Communication Association (INTERSPEECH)*, 2008, pp. 597–600.

[347] M. Wöllmer, M. Al-Hames, F. Eyben, B. Schuller, and G. Rigoll, "A multidimensional dynamic time warping algorithm for efficient multimodal fusion of asynchronous data streams," *Neurocomputing*, vol. 73, no. 1-3, pp. 366–380, December 2009.

[348] M. Wöllmer, B. Schuller, F. Eyben, and G. Rigoll, "Combining long short-term memory and dynamic Bayesian networks for incremental emotion-sensitive artificial listening." *Selected Topics Signal Processing*, vol. 4, no. 5, pp. 867–881, 2010.

[349] D. H. Wolpert and W. G. Macready, "Coevolutionary free lunches," *Evolutionary Computation*, vol. 9, no. 6, pp. 721–735, 2005.

[350] B. Wu, H. Ai, C. Huang, and S. Lao, "Fast rotation invariant multi-view face detection based on real adaboost," in *International Conference on Automatic Face and Gesture Recognition (FGR)*, Washington, DC, USA, 2004, pp. 79–84.

[351] D. Wu, T. D. Parsons, E. Mower, and S. Narayanan, "Speech emotion estimation in 3D space," in *Proceedings of IEEE*, Singapore, July 2010.

[352] W. Wundt, *Principles of physiological psychology*, ser. Principles of physiological psychology, no. Bd. 1. Sonnenschein, 1904.

[353] M.-H. Yang, D. J. Kriegman, and N. Ahuja, "Detecting faces in images: A survey," *Pattern Analysis and Machine Intelligence*, vol. 24, no. 1, pp. 34–58, 2002.

[354] K. Yuen and T. Lee, "Could mood state affect risk-taking decisions?" *Affective Disorders*, vol. 75(1), pp. 11–18, 2003.

[355] M. Zancanaro, B. Lepri, and F. Pianesi, "Automatic detection of group functional roles in face to face interactions." in *International Conference on Multimodal Interfaces (ICMI)*, 2006, pp. 28–34.

[356] Z. Zeng, M. Pantic, G. Roisman, and T. Huang, "A survey of affect recognition methods: Audio, visual, and spontaneous expressions," *Pattern Analysis and Machine Intelligence*, vol. 31, no. 1, pp. 39–58, 2009.

[357] Z. Zeng, J. Tu, B. Pianfetti, M. Liu, T. Zhang, Z. Zhang, T. S. Huang, and S. E. Levinson, "Audio-visual affect recognition through multi-stream fused HMM for HCI," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005, pp. 967–972.

[358] Z. Zeng, Y. Hu, Y. Fu, T. S. Huang, G. I. Roisman, and Z. Wen, "Audio-visual emotion recognition in adult attachment interview," in *International Conference on Multimodal Interfaces (ICMI)*, New York, NY, USA, 2006, pp. 139–145.

[359] Z. Zeng, J. Tu, B. M. Pianfetti, and T. S. Huang, "Audio-visual affective expression recognition through multistream fused HMM," *Multimedia*, vol. 10, no. 4, pp. 570–577, June 2008.

[360] C. Zhang and Z. Zhang, "A survey of recent advances in face detection," Technical report, Microsoft Research, Technical Report, 2010.

[361] S. Zhang, Q. Tian, S. Jiang, Q. Huang, and W. Gao, "Affective MTV analysis based on arousal and valence features," in *International Conference on Multimedia and Expo (ICME)*, June 2008, pp. 1369–1372.

[362] Z. Zhang, E. Coutinho, J. Deng, and B. Schuller, "Cooperative learning and its application to emotion recognition from speech," *Audio, Speech, and Language Processing*, vol. 23, no. 1, pp. 115–126, Jan 2015.

[363] X. Zhu, "Semi-supervised learning literature survey," Computer Sciences, University of Wisconsin-Madison, Technical Report 1530, 2005.

[364] M. Zuckerman and R. E. Driver, "What sounds beautiful is good: The vocal attractiveness stereotype," *Journal of Nonverbal Behavior*, vol. 13, pp. 67–82, 1989.